Lecturing Materials Generating Tools (LMaGTools) (Tools to Support Notes Generating Tasks)

Prepared by
Law Yong Sein (WEK 990037)

Supervised by
Cik Nor Aniza

Moderated by
Mr. Ibrahim Abu Bakar

Abstract

Education is always a concern of everyone, especially educator like teachers, lecturers and tutors. To ensure their students learn more efficiently, they need a set of good teaching materials. Learning Material Generating Tool (LMaGTool) is a tool that provides various features to help them produce better teaching materials in shorter time. This tool is designed to operate under Microsoft Office applications environment, which is their familiar environment.

Teaching programming language is more efficient with materials generated by LMaGTool. Codes can be displayed highlighted, and focused block-by-block, line-by-line or word-by-word. A code simulator will help student to understand the code execution more thoroughly.

LMaGTool consist of two parts, content editing tools and content displaying tools. By using content editing tools, users can edit learning materials easier and they can include some dynamic contents in these materials. While editing material content getting more efficient, displaying task also been optimized with special designed displaying features to support dynamic contents prepared by the editing tool.

This project is proposed to develop using Visual Basic for Applications, a language shipped with Microsoft Office that allows office development. All of the features have been building to deal with Office objects. It is believed that this tool will become an efficient tool to generate learning materials.

Acknowledgement

Many individuals have assisted and helped me in preparing my final year project. I would like to express my deepest gratitude and thanks to my project supervisor, Cik Nor Aniza, who has given me an opportunity to do this final year project. Thanks to her valuable advices, support and guidance. I would like to thank Mr. Ibrahim Abubakar, moderator of my final year project, for his encouragement and understanding.

I would also like to thank my friend Khoon Siah, for his perspectives, and being one of the funniest people I know; and Chia Liang, for his encouragement, and being one of the craziest people I know. His ideas are always refreshing. And to Cynthia, for the balance she gave to me.

Table of Content

DECLARATION A

ABSTRACT B

ACKNOWLEDGEMENT C

LIST OF FIGURE H

CHAPTER 1 INTRODUCTION 1

1.1 PROJECT OVERVIEW 1

1.2 PROJECT OBJECTIVE 3

1.3 PROJECT SCOPE 4

1.4 SIGNIFICANT OF PROJECT 5

1.4.1 This tool helps to reduce time spent to create, manage and generate whole set of learning material. 5

1.4.2 Customize Microsoft Office Applications for better utilization of functions. 5

1.4.3 Provides better ways to explain programme code samples. 5

1.4.4 Better timing when handling a lecture. 6

1.5 PROJECT OUTCOMES 7

1.6 PROJECT DEVELOPMENT STRATEGY 8

1.7 PROJECT SCHEDULE 9

CHAPTER 2 LITERATURE REVIEW 10

2.1 WHAT IS THE MOST EFFECTIVE TEACHING METHOD? 10

2.1.1 Algorithm Simulation 10

2.1.2 Problem based learning (PBL) approach 11

2.2 VISUAL BASIC FOR APPLICATIONS 13

2.2.1 What is Visual Basic for Applications 13

2.2.2 Visual Basic for Applications technology issues 13

2.2.3 Strength of Visual Basic for Applications 14

2.2.4 The Office and Visual Basic Relationship 14

2.2.5 How Does Visual Basic for Applications Relate to Visual Basic? 15

2.3 COMPONENT OBJECT MODEL (COM) 18

2.3.1 What is COM? 18
2.3.2 COM and Visual Basic 18
2.4 COM ADD-INS 20
2.4.1 What is COM add-in? 20
2.4.2 Why we need to develop COM add-ins? 20
2.4.3 The Characteristics of an Add-In Model 21
2.5 MICROSOFT OFFICE OBJECT 24
2.5.1 Microsoft Office Objects 24
2.5.2 The Object Model 24
CHAPTER 3 METHODOLOGY **26**
3.1 INTRODUCTION 26
3.2 ANALYSIS OF THE SYSTEM 28
3.3 CONSIDERATION ON THE DEVELOPMENT TOOL 30
3.4 FUNCTIONAL REQUIREMENTS 33
3.4.1 File management 33
3.4.2 New and open file 34
3.4.3 Save and close 35
3.4.4 Print 35
3.4.5 Code sample processing 35
3.4.6 Algorithm animation 36
3.4.7 Drag and drop 37
3.4.8 Convert file format 37
3.4.9 Pop up 37
3.4.10 Displaying progress bar 38
3.4.11 Menu and toolbars 38
3.4.12 Help features 38
3.5 NON-FUNCTIONAL REQUIREMENTS 40
3.5.1 User friendly and ease of use 40
3.5.2 Reliability 40
3.5.3 Efficiency 40
3.5.4 Simplicity 40
3.5.5 Reusability 41
3.5.6 Maintainability 41

3.6 RUN-TIME REQUIREMENT 42
3.6.1 Software Requirement 42
3.6.2 Hardware Requirement 42
CHAPTER 4 SYSTEM DESIGN **43**
4.1 OVERVIEW OF PROJECT ARCHITECTURE 43
4.2 OVERVIEW OF PROJECT ELEMENTS 46
4.3 PROJECT FLOW DESIGN 49
4.3.1 File management: 51
4.3.2 Content Editing 51
4.3.3 Document conversion 52
4.3.4 Help retrieving 52
4.4 INPUT DESIGN 53
4.4.1 Normal input 54
4.4.2 Pop up input screen 54
4.4.3 Code input screen 54
4.4.4 Insert indicator setting panel 55
4.5 OUTPUT DESIGN 56
4.5.1 Pop up output screen 56
4.5.2 Code output screen 56
4.5.3 Progress bar output 57
4.6 USER INTERFACE DESIGN 59
4.6.1 Menu 59
4.6.2 Toolbar 62
4.6.3 Progress bar 62
4.7 HELP DESIGN 63
CHAPTER 5 SYSTEM IMPLEMENTATION **65**
5.1 INTRODUCTION 65
5.2 SYSTEM CODING 66
5.2.1 Visual Basic 66
5.2.2 Reference for the Project 67
5.2.3 Programming Standards 69
5.2.4 Matching Design with Implementation 69

5.3 PROJECT PROGRAMMING 70
5.3.1 Control Structures 70
5.3.2 Algorithms 70
5.3.3 Data Structures 71
5.4 SUMMARY 72
CHAPTER 6 SYSTEM TESTING **73**
6.1 INTRODUCTION 73
6.2 TYPE OF TESTING 74
6.2.1 Objective and Specification Review 74
6.2.2 Unit Testing 74
6.2.3 Integration Testing 75
6.2.4 System Testing 75
6.2.5 Execution of Testing and the Solution 76
6.3 SUMMARY 81
CHAPTER 7 SYSTEM EVALUATION **82**
7.1 INTRODUCTION 82
7.2 PROBLEMS AND SOLUTIONS 83
7.3 EVALUATING BY END USER 86
7.4 SYSTEM STRENGTHS 91
7.5 SYSTEM CONSTRAINTS 93
7.6 FUTURE ENHANCEMENTS 94
7.7 KNOWLEDGE AND EXPERIENCE GAINED 95
7.8 SUMMARY 96
7.9 CONCLUSION 97
BILIOGRAPHY **98**
APPENDIX **99**
USER MANUAL **105**

List of Figures

FIGURE 1.1: SIMPLIFIED FLOW OF STUDENT LEARNING PROCESS....2
FIGURE 1.2: PROTOTYPING MODEL....8
FIGURE 1.3: SCHEDULE OF THIS PROJECT SHOWN IN A GANTT CHART....9
FIGURE 2.1: MENU ITEMS TO ACCESS VISUAL BASIC EDITOR....16
FIGURE 2.2: THE VISUAL BASIC EDITOR....17
FIGURE 2.3: OBJECTS HIERARCHY IN POWER POINT....25
FIGURE 3.1: THE RAPID APPLICATION DEVELOPMENT (RAD) ROUTE....27
FIGURE 3.2: THE VISUAL BASIC 6 ENVIRONMENT WITH MOST WINDOWS OPENED....32
FIGURE 3.3: THE FILE SYSTEM OBJECTS HIERARCHY....34
FIGURE 4.1: PROJECT ARCHITECTURE FLOW DIAGRAM....44
FIGURE 4.2: DECOMPOSITION DIAGRAM OF THE PROJECT....45
FIGURE 4.3 BASIC ELEMENT OF THE PROJECT....48
FIGURE 4.4: PROJECT FLOW DIAGRAM....50
FIGURE 4.5: INPUT SCREEN OF POP UP....54
FIGURE 4.6: CODE INPUT SCREEN....55
FIGURE 4.7: OUTPUT SCREEN OF POP UP....56
FIGURE 4.8: CODE OUTPUT SCREEN....57
FIGURE 4.9: PROGRESS BAR BEEN PLACED AT THE BOTTOM OF A SLIDE....57
FIGURE 4.10: NEW MENU ITEM INTEGRATED INTO EXISTING MENU BAR IN OFFICE.59
FIGURE 4.11: A PULL-DOWN AND CASCADING MENU FROM MENU BAR....60
FIGURE 4.12: MENU STRUCTURE OF THE PROJECT....61
FIGURE 4.13: A FLOATING TOOLBAR DESIGNED FOR THIS PROJECT....62
FIGURE 4.14: A SIMPLE PROGRESS BAR....63
FIGURE 4.15: ELEMENTS OF THE ASSISTANT'S BALLOON....63
FIGURE 5.1: DIALOG FOR USER TO CHOOSE THEIR PROJECT TYPE....67
FIGURE 5.2: REFERENCE FOR THE PROJECT....68
FIGURE 6.1: EVERY BUTTON IS TESTED FOR THEIR FUNCTIONS....77
FIGURE 6.2: USERS INPUT PLAIN CODE IN THE CODE SECTION....79
FIGURE 6.3: USER USE HIGHLIGHT FUNCTION TO HIGHLIGHT THE CODE....80
FIGURE 7.1: SAMPLE SURVEY FORM....87
FIGURE 7.2: SELECTED SURVEY FORM 1....88
FIGURE 7.3: SELECTED SURVEY FORM 2....89

Chapter 1

Introduction

Chapter 1 Introduction

1.1 Project Overview

University of Malaya provides an excellent educational environment with a proud history, a past it have the privilege of being able to reflect on, and a future that it has the honor of shaping. By innovating existing lecturing style an impressing growth in student concentration during learning process is expected.

As Computer Language subjects become a core part of the study of computer science and information technology, this research will concentrate on improving lecturing process of these subjects. There are ways to improve the process but for sure a better tool will greatly increase the effectiveness of the process.

Studying existing lecturing process shows that only some common tools are used to conduct lectures. These tools include Microsoft Power Point, Microsoft Word and hard copy documents. Each of these tools has their own limitation in producing and displaying the learning materials. Hence, this research will focus on improving existing function of these tools besides develop new features on them. With these improved functions, an efficient and effective tool to generate learning materials, Learning Material Generating Tool (L-MaG Tool), will be developed.

This tool would not be successful until some improvement in displaying the generated learning materials were carried out. Thus, a better lecturing style will be taken into consideration. While figuring out the expected outcome of the lecturing style, the research has to work back to build features to display these materials dynamically according to the lecture needs.

By the time the mentioned features been develop, a lecture can be conduct in easier way and thus an improvement in understanding Computer Language among students is expected. Anyhow, a closer look at the whole picture of student learning process again is necessary.

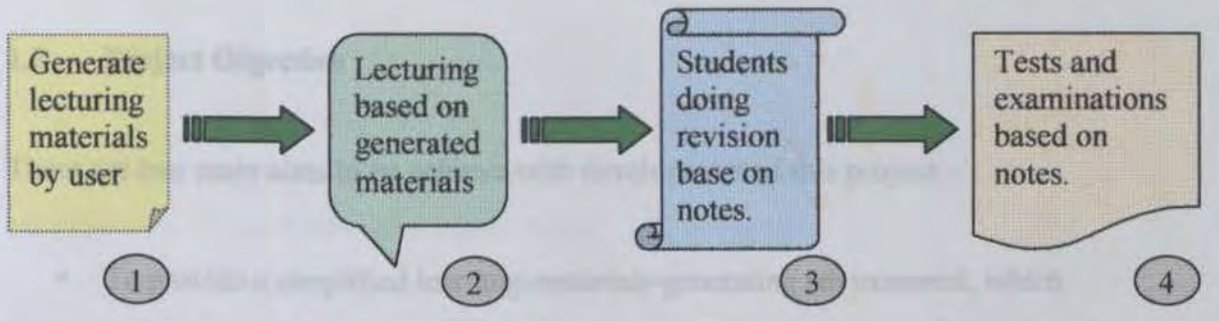

Figure 1.1: Simplified flow of student learning process.

Now the first and second stage of the process had been taken into consideration in the tool development. There are few additional functions in the tool is needed as the reference notes in third stage need to be generated in a different way from notes in second stage, which is only a plain document version of second stage. This is a summary of the lecture's notes. With this second version of simplified notes student can easily download notes whether from course website or print it out to as hard copies of notes for reference.

1.2 Project Objective

There are two main aims to be achieve with development of this project:

- To provide a simplified learning-materials-generating environment, which assist user to create powerful teaching material to support more dynamic and interactive lecturing style.
- L-MaG Tool should be able to generate printable notes from the learning materials.

Before achieving these aims, a number of objectives shall be set as below:

- Integrate L-MaG Tool with Microsoft Office so that it works under Office environment.
- Able to provide user-friendly interface for input and processing.
- Able to shorten time spent to generate learning materials.
- Able to access existing functions in Microsoft Office. Thus users can benefit from both L-MaG Tool and Microsoft Office.
- Able to create interesting material outcome and displaying behavior to ensure drawing attention from students.
- To reduce the complication of using unfamiliar features in Microsoft Office.

1.3 Project Scope

T-MaG Tool is a tool designed to enable user to generate learning materials in the simplest way. Thus, it covered scope as listed:

- Guided user interface - all input interface and file processing interface are simple and all procedure are clearly indicated.
- New set of functions and components of user interface including menu items, toolbar buttons, combo boxes, and edit controls will be integrated into existing Microsoft Office environment, without losing any strength of existing Office functions.
- Users will be able to working their task cross the Microsoft Office applications. For example: Users can initialize their task in Microsoft Power Point but later convert it to Microsoft Word, and also working the other way back. Users have to follow certain formats and ways to input in order to convert their work across these applications.
- Converting learning materials to printable document.

1.4 Significant of Project

There are a few points that encouraged development of this project:

1.4.1 This tool helps to reduce time spent to create, manage and generate whole set of learning material.

Most of the learning materials, which lecturers currently using to conduct their lectures are Microsoft Power Point slide, Microsoft Word documents, and some web-based materials like HTML-coded web pages. To create these materials manually by lecturers is very time consuming. Processes such as typing, formatting, managing existing files and converting them to other displayable format are filling large portion of lecturers' daily schedules.

By developing L-MaG Tool, lecturers can simplified their tasks to generate learning materials. And thus, they will have more time to concentrate in other duties, such as counsel students, conducting lectures and doing researches.

1.4.2 Customize Microsoft Office Applications for better utilization of functions.

Microsoft Office is distributed as Off-the-shelf application software. In this sense Microsoft Office is typically used to support common office activities that do not require any specific tailoring. It is possible to combine the advantages of the L-MaG Tool with Microsoft Office Applications, especially Power Point and Word. By customizing Microsoft Office, users will find many useful functions, both exist in Office and been introduce in L-MaG Tool, become easy to use.

1.4.3 Provides better ways to explain programme code samples.

L-MaG Tool customizes ways of display Power Point slide differently from how it used to be display. For example, samples of programming codes can be display in a panel after a click. And to compare two different codes, another

panel can be display in another click. With both panels displayed side-by-side, any difference between these codes will be easier to compare.

1.4.4 Better timing when handling a lecture.

- L-MaG Tool provides graphical view of progress of a running lecture. A small progress indicator will automatically add into these materials let users know which part of the document they are currently working with. Thus, users know whether they are shall speed up or slow down based on the indicator.

1.5 Project Outcomes

L-MaG Tool focuses on developing a Microsoft Office add-in that can be easily plugged into existing Microsoft Office environment. L-MaG Tool has the following characteristics:

- A specially designed slide interface for Microsoft Power Point.
- An input interface to support automation of generating notes.
- Can work across Office Applications, such as Power Point and Word
- Able to deal with Microsoft Office internal objects.
- It is a user-friendly tool. Users can understand how it works in minutes.
- Provides help features. Users can access to the help sections to get suggestions and explanations about the tool.

1.6 Project Development Strategy

For this project, prototyping model is selected as process model in software engineering literature. Prototyping model uses iterative approach where each system development life cycle (SDLC) is repeated several times (iterated). Requirements and alternative solutions are analyzed in the process. During this process, solutions are also designed and a portion of system is implement and subject to user review. Prototyping model consist of six steps as shown in Figure 1.2.

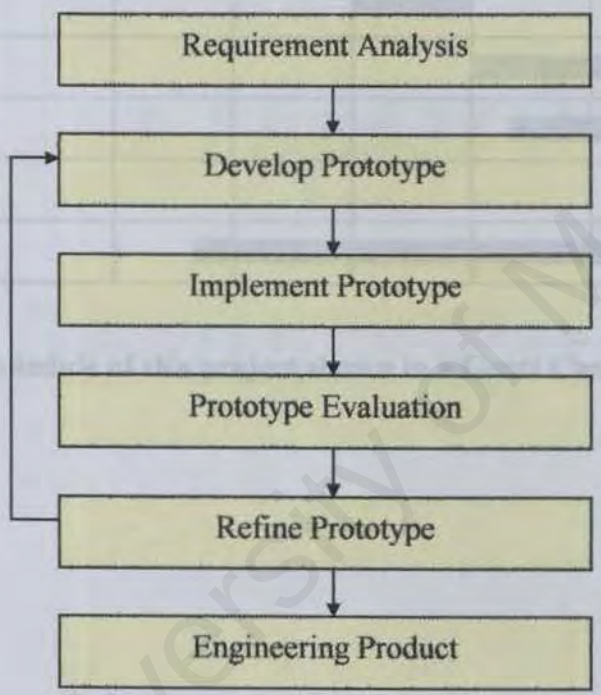

Figure 1.2: Prototyping Model

1.7 Project Schedule

A project schedule describes the software development cycle for a particular project by enumerating the phases or stages of a project and breaking each into discrete tasks or activities to be done. The schedule of this project is shown in Gantt chart below:

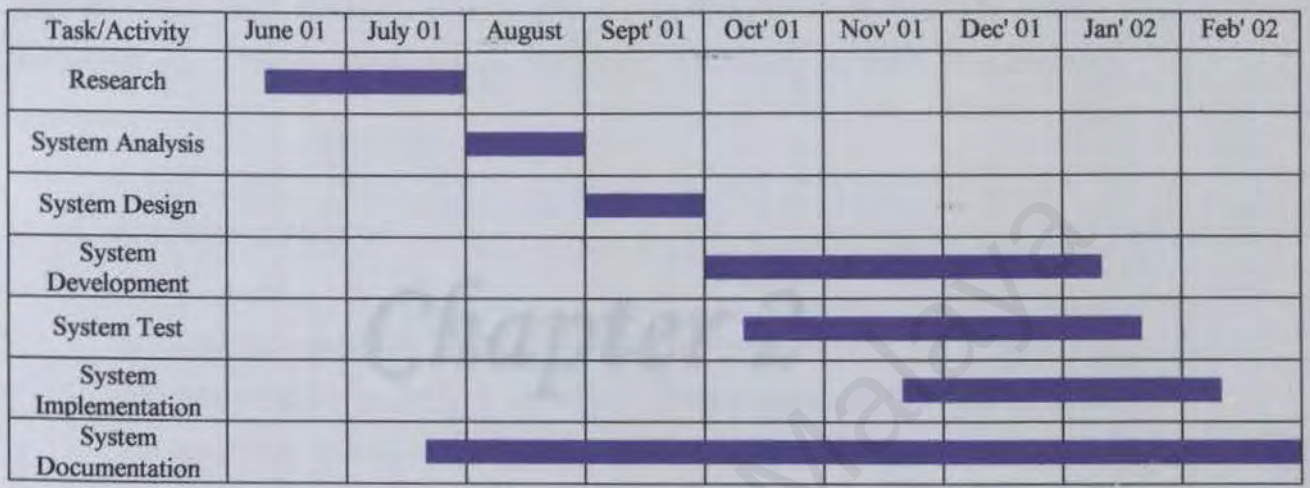

Figure 1.3: Schedule of this project shown in a Gantt Chart

Chapter 2
Literature Review

Chapter 2 Literature Review

2.1 What is the most effective teaching method?

For this tool to function in lecturing process in the most effective way there is one very important study, the study of most effective teaching method, should be made before the any software design process take place. This study included a few teaching methods:

2.1.1 Algorithm Simulation

Ari Korhoren and Lauri Malmi [1] believe that visualization is a useful aid for understanding the working of algorithms. They pointed out that data structures and algorithms belong to the core issues on computer science education, thus student should learn how various data structures are constructed and how different algorithms manipulate them. In addition, the students should gain a proper understanding of the analytical properties of algorithms and their applicability to different practical cases.

There are, however, still have drawbacks for this methods. It is quite possible that some students might take these animations as entertainment without seriously trying to learn from them. Even if this is not true, the student may believe they have understood how the algorithm works, when in practice they have misinterpreting something. Thus, an opportunity shall be given to test their understanding.

An obvious way to test their understanding is "Manual Algorithm Simulation". With this, a simple exercise which students have to demonstrate how a given algorithm manipulates a given data structure step by step.

TRAKLA [2] is a system for teaching basic computer science data structures and algorithms. Some important features of TRAKLA are:

- Distribution of individually tailored algorithm simulation exercise (by email or by Java Applet),
- Accurate and automatic assessment of the exercises,
- Meaningful and immediate feedback about the exercises, and
- Automatic administration of the course.

Students can take advantage of this system to test whether they have understood how algorithms work.

This case shows that a good simulating of algorithms is important in process of understanding how these algorithms work. Thus, the L-MaG Tool will take this approach to simulate algorithm within the teaching materials generated.

2.1.2 *Problem based learning (PBL) approach*

- What is PBL:

  This is a learner-centered approach in which learning episodes are motivated by an initial problem that bears some resemblance to "real world" problems. As such, an important characteristic of a problem is that it be ill structured. Student then proceed to (often iteratively) interpret the problem and work together in small groups to explore potential solutions to it. [3]

- The advantages of PBL:

  The use of problems to simulate student learning in PBL environments led us to a position where we sought to incorporate a selection of problems from which student would choose. This would permit recognition of students' other interests, and give them greater opportunity for "ownership" of the problem, while reflecting the interdisciplinary possibilities of computer science. This allows us to offer problems with the biological "flavors", or an economics one, etc. It also has the advantage of

allowing staff to offer "flavor s" which match their personal interests, and thereby encourage ongoing evolution of the course offerings.

- Drawbacks of PBL:

  Anecdotally, the presence of the conventional stream in parallel with the PBL stream was an initial direction to some students. Coming mostly from high-school backgrounds in which they have adapted to highly structured, teacher-focused learning environments (some quite successfully), there were some students that may find them difficult to deal with the situation given without guidance from lecturer. Some students clearly desired the passive learning experience. This dissipated as students gained more control over the language and started to "buy into" the PBL philosophy.

- Shall we include PBL approach in the tool design?

  As we study the strength and weakness of PBL, we can see it sounds a good practice of teaching and learning. However, it is not a good time to implement in our faculty.

  Due to the highly structured teaching and teacher focused learning environments, which is most secondary schools in Malaysia provide, students are playing a passive role in the learning process. Which means they need to be guide from a step to another before they manage to solve a problem. Introducing this approach should take a lower profile. Which means we shall replace a little portion of course activities with a little PBL approach. This would take longer time to make PBL a workable model of learning in this faculty but decrease the risk, which students find them cannot adapt to the approach.

2.2 Visual Basic for Applications

2.2.1 What is Visual Basic for Applications

One of the primary reasons for the amazing success of Microsoft Office is that Word, Excel, PowerPoint, and the other Office applications share a common tool for customization: Visual Basic for Applications, or VBA. Microsoft's decision to license VBA to Independent Software Vendors (ISVs) has already led more than 150 ISVs to decide to integrate VBA into their own products.

Why have so many software companies licensed VBA? To answer this question, we will need to understand the technology and business issues involved in the decision of whether to integrate Microsoft VBA into our product.

2.2.2 Visual Basic for Applications technology issues

Microsoft VBA is built upon COM, Microsoft's architecture for software components. For programmers to be able to customize their application using VBA, their product architecture must be based on COM. In particular, they will need to design and implement an object model that exposes their application's features using the Automation facilities that are an essential part of COM.

If they have already decided to support a programmable object model based on COM, their next technology decision is whether to include a macro or scripting language in their product. If their customers or business partners require the ability to customize their application, then they will probably need to include a macro development tool such as VBA in their product. A hidden advantage of including VBA in these products is that this allows them to ship a smaller, simpler product to their customers, who can depend on the ability of their services organization or business partners to customize their products to the exact requirements.

Although Microsoft VBA is not the only technology option for ISVs who need a COM-based macro development tool, there are substantial advantages over existing alternatives such as VBScript or programmers' own proprietary scripting language.

2.2.3 *Strength of Visual Basic for Applications*

- Compatibility with Visual Basic.

VBA is built from the same source code base as Visual Basic; thus, providing us with a high level of compatibility. VBA also uses the same high-performance language engine and programmer productivity tools enjoyed by millions of Visual Basic developers.

- Compatibility with Microsoft Office.

As VBA is build into Office 2000, by programming using VBA to build this application, the compatibility of this application with Microsoft Office 2000 is guaranteed. Thus, VBA becomes an easiest way to build an application that fully compatible with Office 2000.

- Superior power and performance.

The Microsoft VBA offers a wide range of options for integrating functions into existing Office environment. These options give us control over every office programmable object. With the access to these programmable objects, we are able to customize a unique Office environment that is suitable for generating learning materials and better display features of these materials. VBA also offers the same outstanding performance found in Visual Basic.

2.2.4 *The Office and Visual Basic Relationship*

Visual Basic for Applications is a combination of an integrated programming environment (the Visual Basic Editor) and the Visual Basic programming

language. This combination allows programmers to easily design and develop Visual Basic programs. The term "for Applications" refers to the fact that the Visual Basic programming language and the development tools in the Visual Basic Editor are seamlessly integrated with all Office applications so that programmers can develop custom functionality and feature solutions using these applications.

2.2.5 *How Does Visual Basic for Applications Relate to Visual Basic?*

Visual Basic for Applications (also known as Visual Basic, Applications Edition) isn't the same as, and shouldn't be confused with, Microsoft Visual Basic. Office features the Visual Basic language and exposes the ability to control Office functionality through a set of programmable objects. Using the Visual Basic Editor and the different objects exposed by Office (which are integrated with all Office applications) you can create specialized programs for Office. Programmers can store these programs in an Office document or in a separate file called an add-in.

The tools and graphical user interface provided by the Editor are consistent with the Microsoft Visual Basic version 6.0 development environment. Visual Basic 6.0, however, provides much more advanced programming tools and functionality, so programmers can create complex programs for the Microsoft Windows operating system and components for other Windows programs. Programmers can develop self-contained executables (.exe files) as well as application extensions (.dll files) for Office using the tools in the Visual Basic 6.0 programming system.

2.2.6 *The Visual Basic Editor in Office Applications*

Each document, workbook, presentation, or database you open in Word, Excel, PowerPoint, or Access respectively has an associated Visual Basic for Applications project. When you open a workbook in Excel, for example, an associated Visual Basic project is listed in the Project Explorer window of the Visual Basic Editor. To write Visual Basic code in the Excel workbook's

Visual Basic for Applications project, you must display the Visual Basic Editor. One way to display it is to point to Macro on the Tools menu, and then click Visual Basic Editor on the submenu. Figure 2.1 illustrate menu items to access the Visual Basic editor.

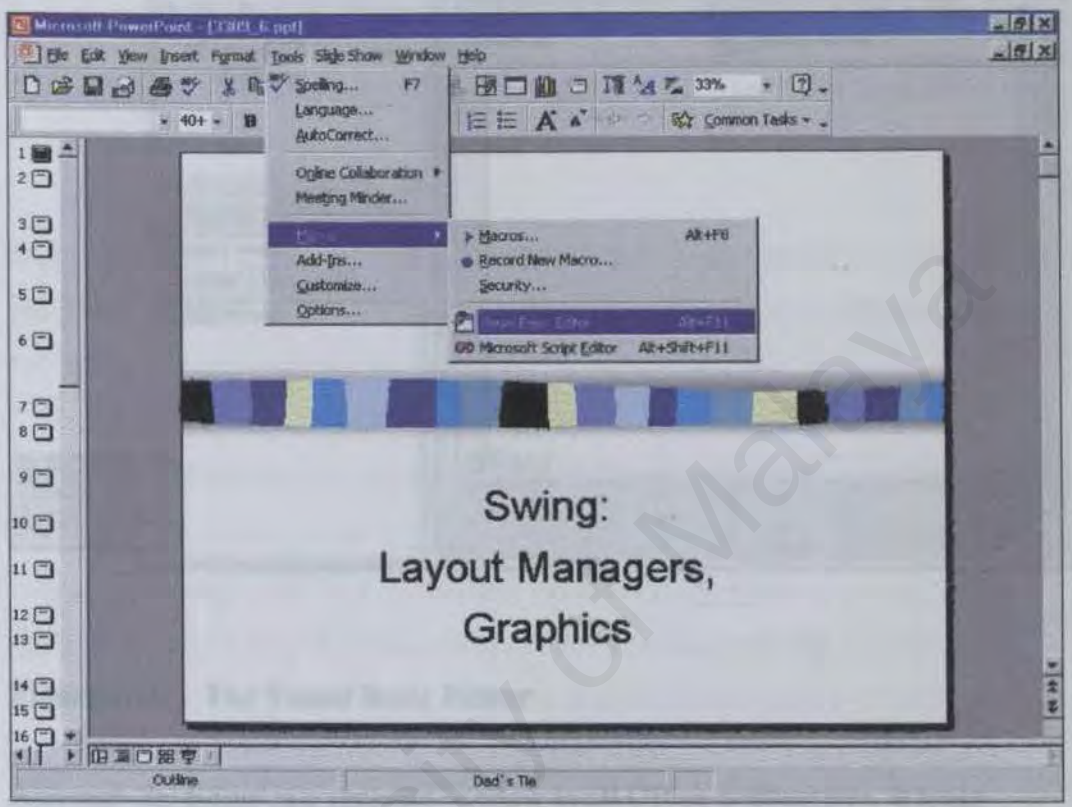

Figure 2.1: Menu items to access Visual Basic editor.

All the applications in Office provide the same integrated development environment. The Visual Basic Editor contains all the programming tools you need to write Visual Basic code and create custom solutions. For example, you can switch to the Visual Basic Editor window from Word the same way you do from Power Point (from the Tools menu, point to Macro, and then click Visual Basic Editor). Figure 2.2 shows the Visual Basic Editor accessed from Power Point.

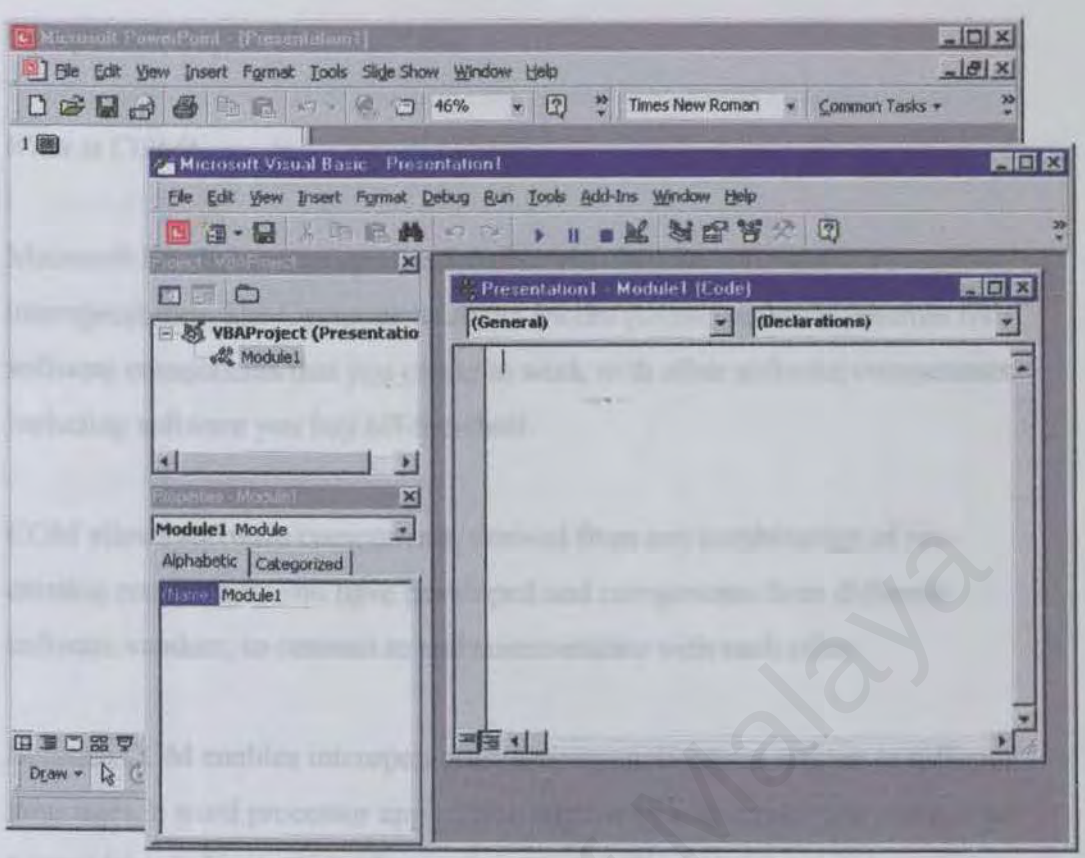

Figure 2.2: The Visual Basic Editor

Although the Editor is a separate window in all Office applications, it looks and functions identically in each one. Thus, you can potentially have three Visual Basic Editor windows open at one time, each one associated with a separate application. When you close a given application, its associated Visual Basic Editor window closes automatically.

2.3 Component Object Model (COM)

2.3.1 What is COM?

Microsoft has defined an open, extensible standard for software interoperability. The Component Object Model (COM) makes it possible for software components that you create to work with other software components, including software you buy off-the-shelf.

COM allows software components, derived from any combination of pre-existing components you have developed and components from different software vendors, to connect to and communicate with each other.

Because COM enables interoperability among applications written in different languages, a word processor application written by one vendor can connect to a spreadsheet object written by another vendor. The word processor application could then be used to import data from the spreadsheet object. Additionally, the spreadsheet object could receive data through a COM object residing on a mainframe. The word processor, spreadsheet, and mainframe database do not have to know anything about each other's implementation. The word processor only needs to know how to connect to the spreadsheet; the spreadsheet only needs to know how to expose its services for other software components to connect.

2.3.2 COM and Visual Basic

As you define the functionality required by your application, look for places where you can use pre-existing COM components instead of having to write the code to implement a given functionality yourself. For example, you could create a function procedure that logs a user into a server. In your corporation, you may be required to include this procedure in every application you create. Instead of adding this code to every program, you can implement this logon procedure in a COM component that can be easily reused from project to project.

There are many reasons to use COM components in an application, including:

- Reusability

Once you create a COM component, other developers can use it. This enables easy access to your component's features in other applications without requiring developers to write extensive code.

- Reduced complexity

You can create a COM component to hide programming complexity from other programmers. Other programmers need only know what information to provide to your component, and what information to retrieve.

- Easier updating

Components make it easier for you to revise and update your applications. For example, you can create a COM component that encapsulates business rules. If the business rules change, you update just the component, not all the applications that use the component.

2.4 COM Add-Ins

2.4.1 *What is COM add-in?*

Add-ins are tools that you can create to customize and extend the functionality of Word, Excel, PowerPoint, or Access themselves. An add-in is a supplemental program that adds custom commands and specialized features to any Office application. For example, you can write an add-in program for Word, Excel, and PowerPoint that displays a list of contacts retrieved from your Outlook Contacts folder. You can create an add-in in the form of a wizard that steps a user through a series of tasks. Most add-ins are accessed through a menu command or toolbar button.

2.4.2 *Why we need to develop COM add-ins?*

If you buy a hair dryer in Malaysia and you travel to Europe, you will probably need to buy an adapter so you can plug it in and use it there. The hair dryer can't plug in and work in all areas of the world.

In Office 97, an add-in that plugs into Microsoft Word can't readily be plugged into Microsoft Excel or any other Office application. Programmers need to develop an "adapter" to plug an add-in for one Office 97 application into another. In Office 2000, the COM add-in model provides the universal adapter to plug any add-in into any Office application.

The COM add-in model supersedes the capabilities of the add-in model provided by each Office 97 application. Because the COM add-in model supersedes Office 97 capabilities, if programmers have developed add-ins for Office 97, they will have the option of using the COM add-in model in Office 2000 without sacrificing functionality.

2.4.3 *The Characteristics of an Add-In Model*

The following points list the characteristics that make an add-in model for Office successful. Each characteristic is available in the COM add-in model for Office 2000 and is not consistently available in Office 97. These characteristics all provide reasons why you should use the COM add-in model to create add-ins in Office 2000.

- Teaches you how to write an add-in once and apply skills to any application.

Each Office 97 application, except Outlook, has at least two ways to extend its capabilities through add-ins. In Office 2000, when you learn how to create a COM add-in once, you've learned how to create it for any Office application.

- Allows you to write code in one file and load it into any Office application.

The registration of a COM add-in indicates which Office application the add-in file can be loaded into.

- Add-ins are easily portable to other Office applications

Using the COM add-in model, you can add an additional Select Case statement where applicable to distinguish which application the add-in is loaded into.

- Allows you to use the programming language or developer tool you like

When Office loads a COM add-in, it doesn't know what programming language was used to build the add-in file. In Office 97, each application connects and disconnects an add-in in a different way based on the programming language the add-in is developed in.

- Can set an add-in's connection to optimize performance

Word and Microsoft PowerPoint don't provide a way to load an add-in on demand (that is, when a menu or toolbar customization is clicked). Excel and Microsoft Access do provide a way to load an add-in on demand, but in inconsistent ways. COM add-ins provide a consistent way to connect an add-in at application startup or when a user clicks a menu or toolbar customization.

- Allows a registering of add-ins in the Microsoft Windows Registry that's consistent across applications

Each Office 97 application provides a different way and a different Windows Registry key to register an add-in. The keys to register a COM add-in are consistent in each Office 2000 application.

- Provides a consistent and simple way to communicate between two add-ins PowerPoint's

In Office 97, Word, Excel, PowerPoint, and Access each provide the Run method on the *Application* object. However, method varies slightly. With COM add-ins, you can communicate between add-ins consistently using the Object property on the *COMAddIn* object.

- Shows one way to learn how to package and deploy an add-in

The Package and Deployment Wizard, available in Visual Basic 6.0 or Microsoft Office 2000 Developer, provides a consistent way to package and deploy a COM add-in.

- Can create add-ins that know if an Office application is started from the Windows Start menu or Windows Explorer, through Automation using *CreateObject*, or through an embedded object

A value from 1 to 3 is assigned to the first value of the *custom()* array argument passed into the *OnConnection* procedure in a COM add-in, indicating how the Office application is started. Add-ins in Office 97 do not provide a way to determine how the application is started.

- Allows you to determine if an add-in is loaded when the application is started or after it's started

By using the *ConnectMode* argument passed into the *OnConnection* procedure in a COM add-in, you can determine when the add-in is loaded.

- Gives you the ability to create a template for use in any new add-in projects

You can create a template for Visual Basic 6.0 or Visual C++ to reuse every time you create a COM add-in.

- Provides in each application a consistent add-ins dialog box that displays a list of available add-ins

You can access the add-ins dialog box by clicking the Add-Ins menu item or a related item on the Tools menu in each application. In Office 2000 you can access the COM Add-Ins dialog box from the COM Add-Ins command.

2.5 Microsoft Office Object

2.5.1 Microsoft Office Objects

Objects that we perceive by vision or touch surround us all. We distinguish things by their properties, how they are related to other objects, and how they are affected by an action. Most objects have some sort of functional or aesthetic purpose, and many are collections of objects grouped together. An obvious example would be your computer. It consists of a monitor, a keyboard, speakers, a processor, disk drives, a mouse, and perhaps other components. Each component is further composed of objects, until we finally reach a fundamental element.

Software provides a similar ordering of objects. We can't, of course, put them on our mantel, but we can distinguish them by their properties and the relationships they have with one another. In Office, almost all the functionality we work with and all the viewable content we create is represented by an equivalent object in Visual Basic for Applications. Because these objects are programmable, we can develop a Visual Basic program that manipulates the properties that an object exposes. The collections of Office objects are categorized by either Word, Excel, PowerPoint, Access, Outlook, or Office, and they allow us to navigate down to the smallest detail of information in any of our documents.

2.5.2 The Object Model

Most objects are described in relation to another object. For example, a key on our keyboard doesn't stand on its own; it's a functioning part of the whole keyboard, which in turn is part of our computer. It's the relationship of objects that forms the basis of an object model in Office; the model is the hierarchy of objects in relation to each other.

All Office applications have the same general hierarchy model of objects, with the *Application* object residing at the top. Each *object* represents an element of

an application, such as a shape on a slide, a cell in a worksheet, a word in a document, or a table in a database. Navigating up and down the object model hierarchy is similar to using a road map, which displays the routes you might take to reach certain destinations.

The *Application* object represents the starting point. From the *Application* object, we travel down a branched highway, selecting objects to pass through until we reach the object we want. If we want to change the color of a shape in a PowerPoint slide, we start with the *Application* object (PowerPoint), indicate which presentation the slide belongs to, and then find the slide that contains the shape. Finally, we reach our destination by selecting the shape on the slide. Figure 2.3 illustrates the concept of Objects in Microsoft Office Power Point.

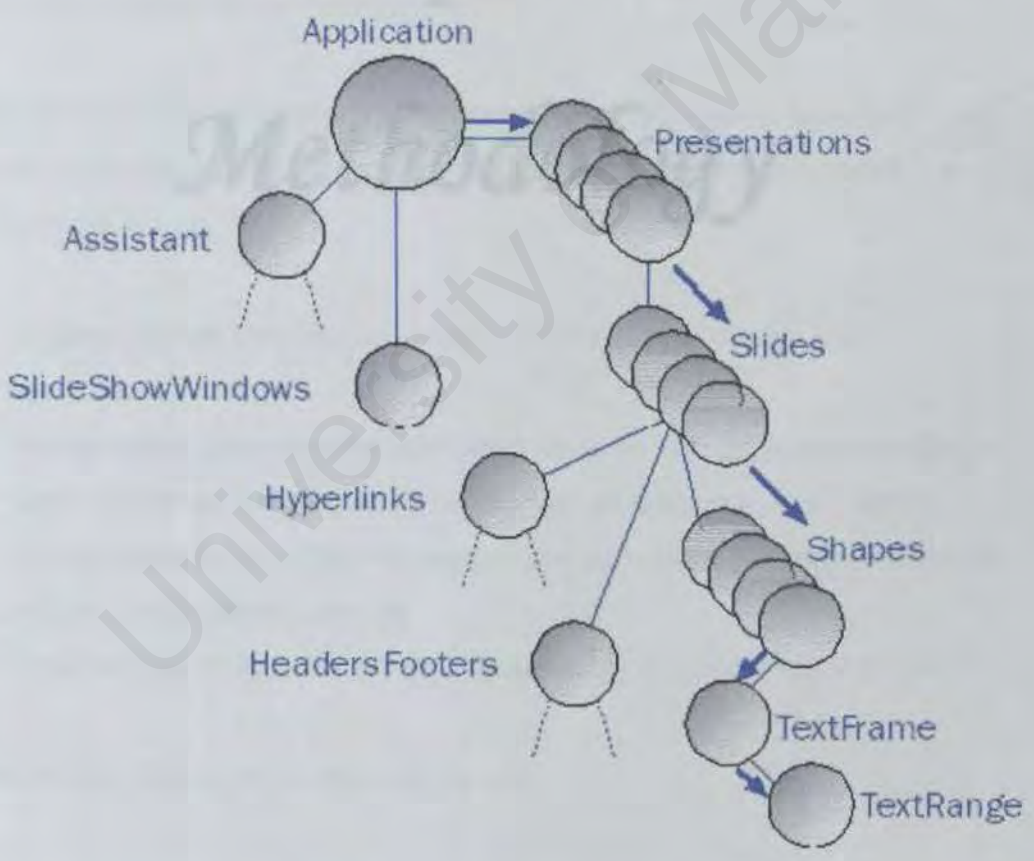

Figure 2.3: Objects hierarchy in Power Point.

Chapter 3

Methodology

Chapter 3 Methodology

3.1 Introduction

System development methodology is a very formal and precise system development process that defines a set of activities, methods, best practices, deliverables and automated tools for system developments and project managers use to develop and maintain most or all information systems and software.

A methodology is a problem solving approach to build systems. The term problems is used in this chapter include real problems, opportunities for improvement and directives from management. (Berjamin, 1997)

The methodology used to develop and support the proposed system is the most commonly used methodology, Rapid Application Development (RAD) route. The basic ideas of RAD are:

- To more actively involve system users in the analysis, design, and construction activities
- To organize systems development into a series of focused, intense workshops jointly involving system owners, users, analysts, designers, and builders
- To accelerate the requirements analysis and system design phases through an iterative construction approach
- To reduce the amount of time until the users begin to see a working system

The reasons why this approach been chosen are:

- It is most popular for smaller system projects
- Users and management see working, software-based solutions more rapidly then in model-driven development.

- It is less risky in the sense of, users and developers can test the system before it take place. Thus, if any failure occurs, users and developers can detect it immediately
- It applies the idea of "divide and conquer", which allow developers deal with a smaller piece of problems one by one
- Projects have higher visibility and support because of the extensive user involvement throughout the process
- The iterative approach is a more "natural" because change is an expected factor during development

Rapid application development is illustrated in Figure 3.1.

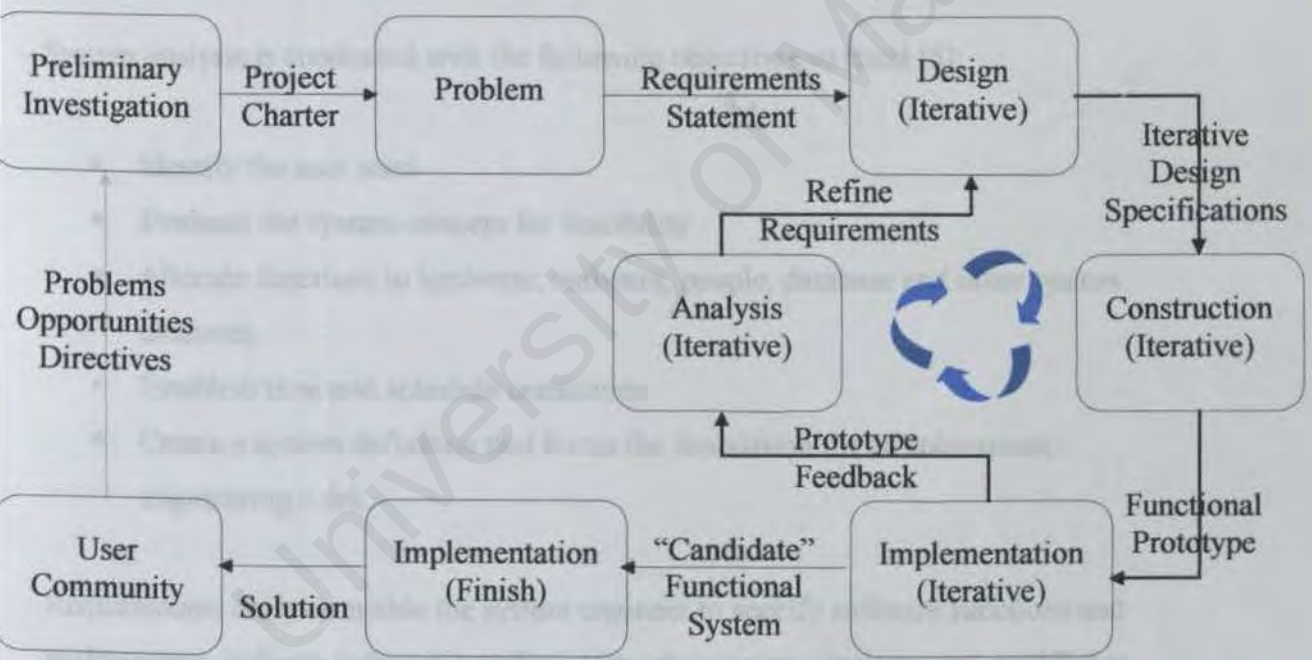

Figure 3.1: The Rapid Application Development (RAD) route.

3.2 Analysis of the system

System analysis is a study of

- Current business and information system application,
- The definition of user requirements, and
- Priorities for a new or improved application.

System analysis is driven by system users' concerns; hence, it addresses the People, Data, Activities and network building blocks from a user perspective [4]. The overall emphasis of analysis is to gather data and the feasibility of these solutions. A complete understanding of software requirement is essential to the success of a software development effort.

System analysis is conducted with the following objectives in mind [5]:

- Identify the user need
- Evaluate the system-concept for feasibility
- Allocate functions to hardware, software, people, database and other system elements.
- Establish time and schedule constraints
- Create a system definition that forms the foundation for all subsequent engineering work.

Requirements analysis enable the system engineer to specify software functions and performance, indicate software interface with other system elements and establishes design constraints that the software must meet. A study of function had been carried out on current available features in Microsoft Office 2000 and other existing toolkits. This study shows that users can perform almost every office activities with Microsoft Office 2000, including typing, document editing, slides editing, scheduling and other. Microsoft Office is a powerful office automation software, but because it is design for general use by most of the users, it doesn't really able to fulfill everyone's need in supporting their work.

Understanding this fact, Microsoft has leaved the door open to public users to customize their own desired Office. To allow users customizing their own functions and features, which they found useful to automate their daily works, Microsoft simply constructed all office elements into programmable objects. By programming these objects, the behavior of the Office applications can be controlled.

3.3 Consideration on the development tool

Before constructing and tailoring the Office applications, it is very important to choose right development tools. These tools must able to:

- Improve productivity – through automation of tasks
- Improve quality – because automated tools can check for completeness, consistency, and contradictions
- Better and more consistence documentation – because the tools make it easier to create and assemble consistent, high-quality documentation
- Methodologies that really work – through rule enforcement and build-in expertise

The emphasis on speed and quality in software development has resulted in RAD approaches. The potential for RAD has been amplified by the transformation of programming language compilers into complete application development environments. Thus, several software development tools had been developed to assist project development in by this approach. These tools include **Application Development Environments (ADE)**, which will take place in our project development process.

Application development environment, also known as Integrated Development Environment (IDE), are integrated software development tools that provide all the facilities necessary to develop new application software with maximum speed quality. ADEs make prototyping and RAD possible; however, ADEs are not exclusive to RAD techniques. Most programming language compilers are now integrated with a full ADE. Some of these well-known language compilers are:

- Allaire's *Cold Fution* (for Web application development)
- IBM's *Visual Age* product family (*C++*, *Smalltalk*, *Java* and others)
- InPrise's *Delphi* and *J builder* (*Java*)
- Microsoft *Visual Studio* (*Visual Basic*, *C++*, and *Java*)
- Microsoft *Access* (*SQL* and *Visual Basic for Applications*)
- Oracle's *Designer 2000*

- Sybase's *PowerBuilder*
- Symentec's *Visual Café* (*Java*)

From the list of language compilers, *Visual Basic* and *Visual Basic for Applications* are selected as the tool to develop the project.

Application development environment provide a number of productivity and quality management facilities. The ADE vendor provides some of these facilities. Third-party vendors provide many other facilities that can integrate into the IDE.

- *Programming languages or interpreters* are the heart of ADE. Powerful debugging features and assistance are usually provided to help programmers quickly identify and solve programming problems.
- *Interface construction tools* help programmers quickly build the user interfaces using a component library
- *Middleware* is software that helps programmers integrate the software being develop with various databases and computer networks
- *Testing tools* are used to build and execute test scripts that can consistently and thoroughly test software
- *Version control tools* help multiple programmer teams manage multiple versions of a programme, both during development and after implementation
- *Help authoring tools* are used to write online help systems, user manuals, and online training
- *Repository links* permit the ADE to integrate with *Computer-aided system engineering (CASE)* tool products as well as other ADEs and development tools

General ADE interface is illustrated in Figure 3.2

Generally these features provided by ADEs are most desired for development of this project. After considered a few available ADEs, Microsoft *Visual Basic for Applications* appears to be the most suitable tool to utilize for the rest of the development process. As mentioned earlier, VBA has a totally same syntax with *Visual Basic* but it provides a better way to access, modify, control, and implement

objects in Microsoft Office. Foreseeing this project is potentially has a large portion of development process dealing with Microsoft Office objects; VBA is no doubt the best ADE to use.

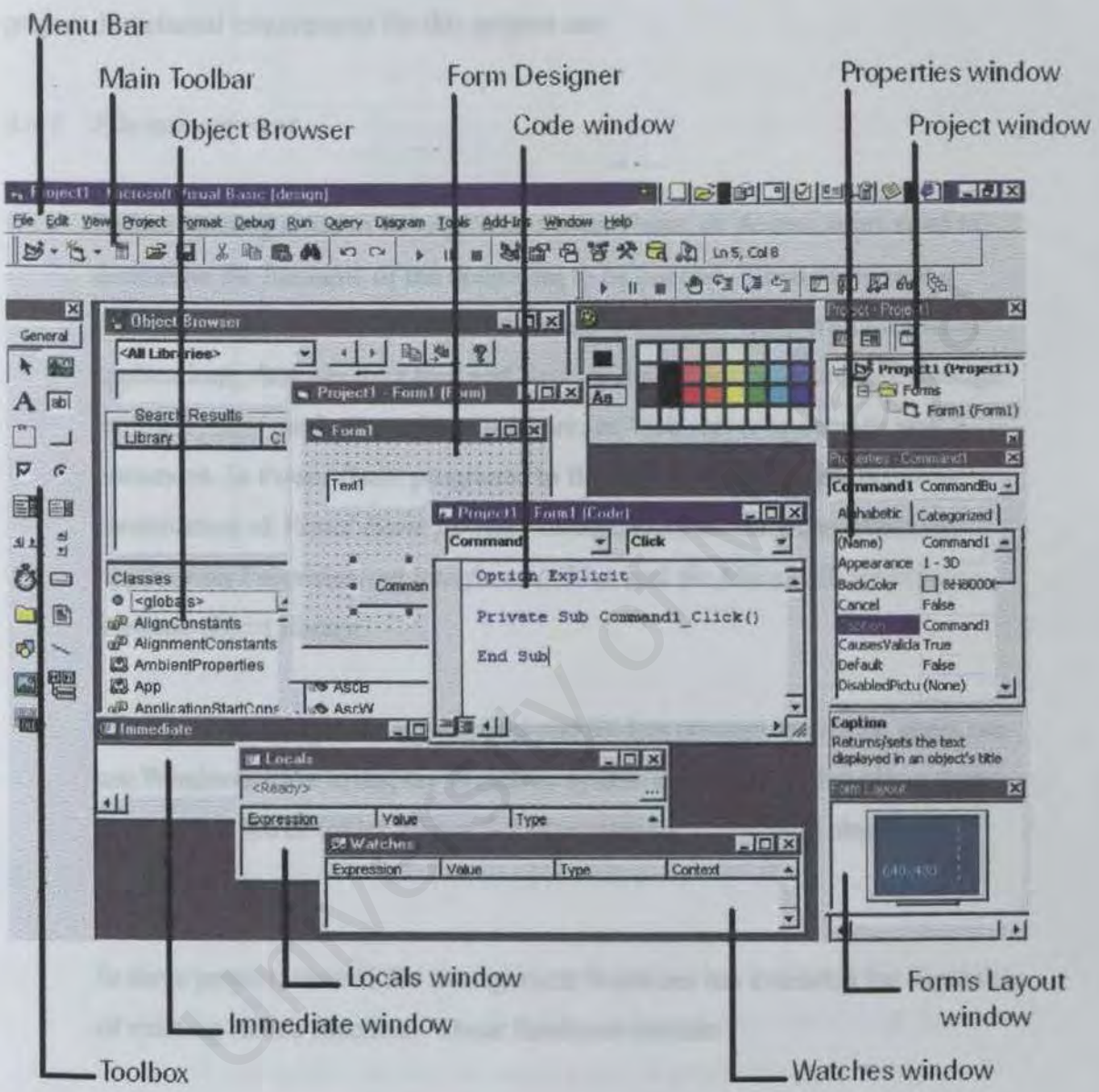

Figure 3.2: The Visual Basic 6 environment with most windows opened.

3.4 Functional requirements

A functional requirement is a description of activities and services a system must provide. In other words, it states a set of required functions, which included into a project. Functional requirement for this project are:

3.4.1 File management

Before opening a file in *Word*, *Excel*, *Power Point*, or *Access*, users need to determine the filename of the document to be opened. When users save a document, they need to specify a filename. When they work with the Office applications, they use the *Open* and *Save As* dialog boxes to navigate through the file system on their machine, network, or Web server to open or save a document. In *Visual Basic* programs, to find files and folders users can use a combination of *Visual Basic for Applications* built-in functions, Windows *Application Programming Interfaces (APIs)*, and the Microsoft *Scripting Runtime* object library.

In addition to using code to find files and folders programmatically, users can use Windows APIs to display Windows system dialog boxes that allow them to specify a file or folder. Figure 3.3 illustrates the file system objects hierarchy.

In these project, several file management functions has extended the capability of existing Office functions. These functions include:

- Check if a file exist
- Check if a folder exist
- Remove a file
- Browse for a file
- Browse for a folder

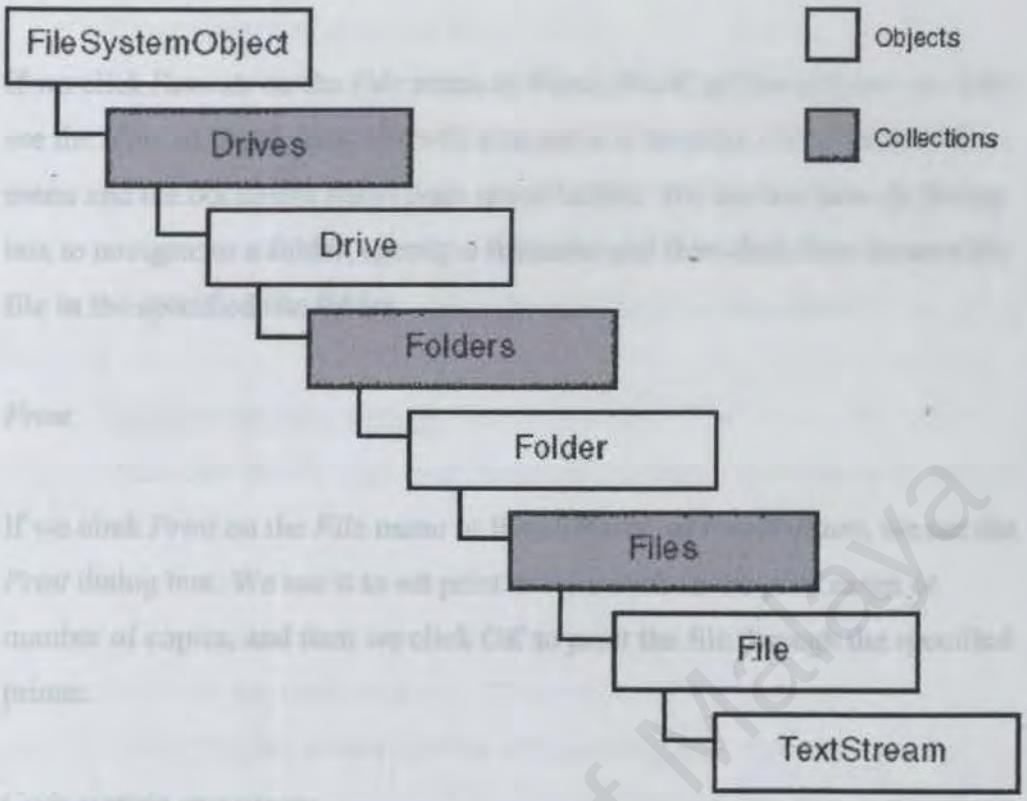

Figure 3.3: The file system objects hierarchy.

3.4.2 *New and open file*

The most common thing we do in Office applications like *Word*, *Excel*, and *Power Point* is to specify an area in a document and then manipulate the content. If we want to create content in a new document or in an existing document that isn't currently loaded into the application, we first need to create a new file or open an existing one. This necessity isn't limited to *Word*, *Excel*, and *PowerPoint* documents. We may also need to create new e-mail messages in *Outlook*, for example, or open an existing database in *Access* to retrieve data for a report in *Word*.

3.4.3 *Save and close*

If we click *Save As* on the *File* menu in *Word*, *Excel*, or *PowerPoint*, we will see the *Save As* dialog box. We will also see it if we click *Close* on the *File* menu and the document hasn't been saved before. We use the *Save As* dialog box to navigate to a folder, specify a filename and then click *Save* to save the file in the specified file folder.

3.4.4 *Print*

If we click *Print* on the *File* menu in *Word*, *Excel*, or *Power Point*, we see the *Print* dialog box. We use it to set print criteria such as the print range or number of copies, and then we click *OK* to print the file through the specified printer.

3.4.5 *Code sample processing*

- Code input

  Provides an input panel, where users can specified the title, purpose, and other information about the code. With this panel, users can input their code sample, whether by typing, cut and paste, drag and drop, or import from other files.

- Code formatting

  Code after the input session will be forward to another panel to perform editing and formatting. These formatting steps include:

  ✓ Format variable

  Variable will be formatted to distinguish them from other part of the code. Varying the colours of the variable is one of the ways. To simplify this formatting task, the tool will automate these actions, while users just have to define the name of the variable once.

- ✓ Format structure of the code

  Each procedure or control structure of the sample programme code is a block of code. From an opening parenthesis to the appropriate closing parenthesis, a block of code can be display separately for better visualization. With these specially designed display options, the code will be easier to handle and explain in the lecturing session.

- ✓ Format displaying setting

  Users can specify their preference on displaying the code in an options panel. They should be able to choose

  - × Font attributes of the text,
  - × Colour of different element, such as variable, control keywords in the code, and
  - × Display characteristics of the code panel.

- Code display

  This tool can provide users some runtime display options. Such as

  - ✓ Whether to be display on top of other windows
  - ✓ Whether two sets of codes (for compare purpose) are display in tile windows or in a single window. And if it is displaying in a single windows, a separator would vertically or horizontally separate the window into two sections.

3.4.6 *Algorithm animation*

Algorithm animation is a function provided by this project to support the explanation of programming algorithm, simple programming code, or block of programming code.

It help lecturers to explain codes in a better way since student will have better visualization of executing the algorithm or code. This feature can avoid inability of student to imagine the execution of certain algorithm or code.

3.4.7 *Drag and drop*

To make editing and input of materials easier, this tool should able to utilize drag and drop idea. In other words, users can simply highlight text they want to put into their material, and drag the selected area to their material editable text field.

3.4.8 *Convert file format*

The tool needs to be able to convert format of a file. Users create, edit and save their material in *Power Point*. By using this convert function, they don't have to retype and reedit the material in *Word*. The tool will automatically rearrange all slides content and save it into *Word* format. This helps users to avoid double work on the same thing, and thus, saving their time.

3.4.9 *Pop up*

- Error messages

  Error massage will inform users for using functions of the tool in improper ways. This error message will provide information and guide for solving the problems users facing.

- User defined messages

  Users can manually customize a pop up information themselves. This little function will increase flexibility to arrange different ways to present learning materials.

3.4.10 *Displaying progress bar*

This function allows users to insert a progress bar into each page or each slide of their materials. Progress bar is the simplest way to indicate the progress of a lecture. This feature helps lecturers to manage their time more effectively.

3.4.11 *Menu and toolbars*

Understanding that this tool will provides numbers of functions; menu and toolbars are necessary for easier access to most of the functions. Newly added menu items will reside on the existing Office menu bar. All related menu item would be group under one main menu item for better recognition.

Graphical toolbars provides fastest access to frequently used functions, and because these toolbars are graphical, it is easy to know what kind of function they link to. Moreover, tool tips function is provided along with the toolbars. This combination of user-friendly features will greatly reduce time spend to understand function of each toolbars.

3.4.12 *Help features*

- Help files

  Help features are accessible from the menu and toolbar. It provides information needed by users to solve problems or learn more about the tool.

- Tool tips

  Tool tips will automatically display when users place their mouse cursor at the toolbar. It makes the tool more user-friendly and easier to understand.

- Office assistant

The Office *Assistant* provides a common interface for displaying *Help* information and tips to users who are working with any Office application. By using the *Assistant* in this tool, it provides *Help* tips that explain how to use features in the tool. It can also display a set of ways to get more information. Since the objects, methods, and properties associated with the Office *Assistant* are defined in the Microsoft *Office 9.0 Object Library*, we can write code that can be shared among, and executed in, all the Office applications without having to tailor the code to each one.

3.5 Non-functional requirements

Non-functional requirements are the constraints under, which software must operate and standards that must be met by the software. That is, a non-functional requirements or constraints describes a restriction on the software, that guide us to find better choices for constructing a solution to the problem. These constraints usually narrow our selection of language, platform, implementation techniques or implementation tools.

3.5.1 User friendly and ease of use

This project will construct a user-friendly functions and interface for the tool. These include graphical user interface (GUI) techniques, simplified steps and functions, and help features to guide users.

3.5.2 Reliability

The tool is designed for reliability. It is expected not causing any unnecessary and undesired downtime of the overall environment. A tool is reliable only if it does not dangerous and costly failures when users utilize it in a proper way. This definition recognizes that a tool may not always be used in the way the designer expects.

3.5.3 Efficiency

Efficiency in computer terminology means a procedure hat can be called or accessed in an unlimited numbers of times to produce similar outcomes or output at creditable speed.

3.5.4 Simplicity

Simplicity refers to keep forms and screens properly uncluttered in a manner that focuses the user attention.

3.5.5 *Reusability*

The extent to which a program or part of it can be reuse in other application. The project is actually extents most of the objects in Office, and is construct by creating and controlling reusable object in Office. These objects and procedure are reusable by future Office developing engineer.

3.5.6 *Maintainability*

A software is maintainable if the functions in the software are easy to modify and test when updating to meet requirement, correcting errors, or move to a different computer system. This tool is mostly constructed based on the Office development strategies. From this point of view, it keeps a very good environment for future maintenance and upgrade.

3.6 Run-time requirement

3.6.1 Software Requirement

This tool's final form will be the Microsoft Office 2000 applications COM add-in. At the mean time, it is still not a stand-alone programme. For this reason, the software requirement is actually simple: All computers running Microsoft Office 2000 will be able to utilize the functions in the add-in.

3.6.2 Hardware Requirement

The hardware requirement consideration is basically not much different from the software requirement, in the sense that we just need a set of compatible hardware that runs Microsoft Office 2000 to operate this tool. Below are the minimum hardware requirements:

Processor	:	Intel Pentium II 500 MHz (or 100% compatible)
Memory	:	64 MB RAM for Windows 95/98
Storage	:	50MB Hard disk space
Software	:	Microsoft Office 2000
		Turbo C++ 3.0 (optional)
Monitor	:	SVGA Colour monitor
Input Devices	:	Keyboard and mouse

Chapter 4
System Design

Chapter 4 System Design

System design is defined as those tasks that focus on the specification of a detailed computer-based solution. There are many strategies or techniques for performing systems design. They include *modern structured design, information engineering, prototyping, JAD, RAD,* and *object-oriented design.* These strategies are often viewed as competing alternative approaches to systems design, but in reality, certain combinations complement one another.

In this project, *Rapid Application Design (RAD)* is used as the design strategy. RAD is the merger of various structured techniques (especially the DATA-driven information engineering) with *prototyping* techniques and *joint application development* techniques to accelerate systems development.

4.1 Overview of project architecture

This project is designed based on architecture as shown in Figure 4.1. This tool is build to customize and implement numbers of handy functions to Office applications. Users can access these functions only from Office applications, which install this add-in tool. Thus, from the illustrated flow of activities, it is clear that this tool is not stand-alone software. Its functions can be utilized to allow new forms of inputs and outputs, which differ from Office applications documents.

More technical information is load into Office documents when the tool is use. And later this information will represent various outcomes to assist lecturers in their lecturing.

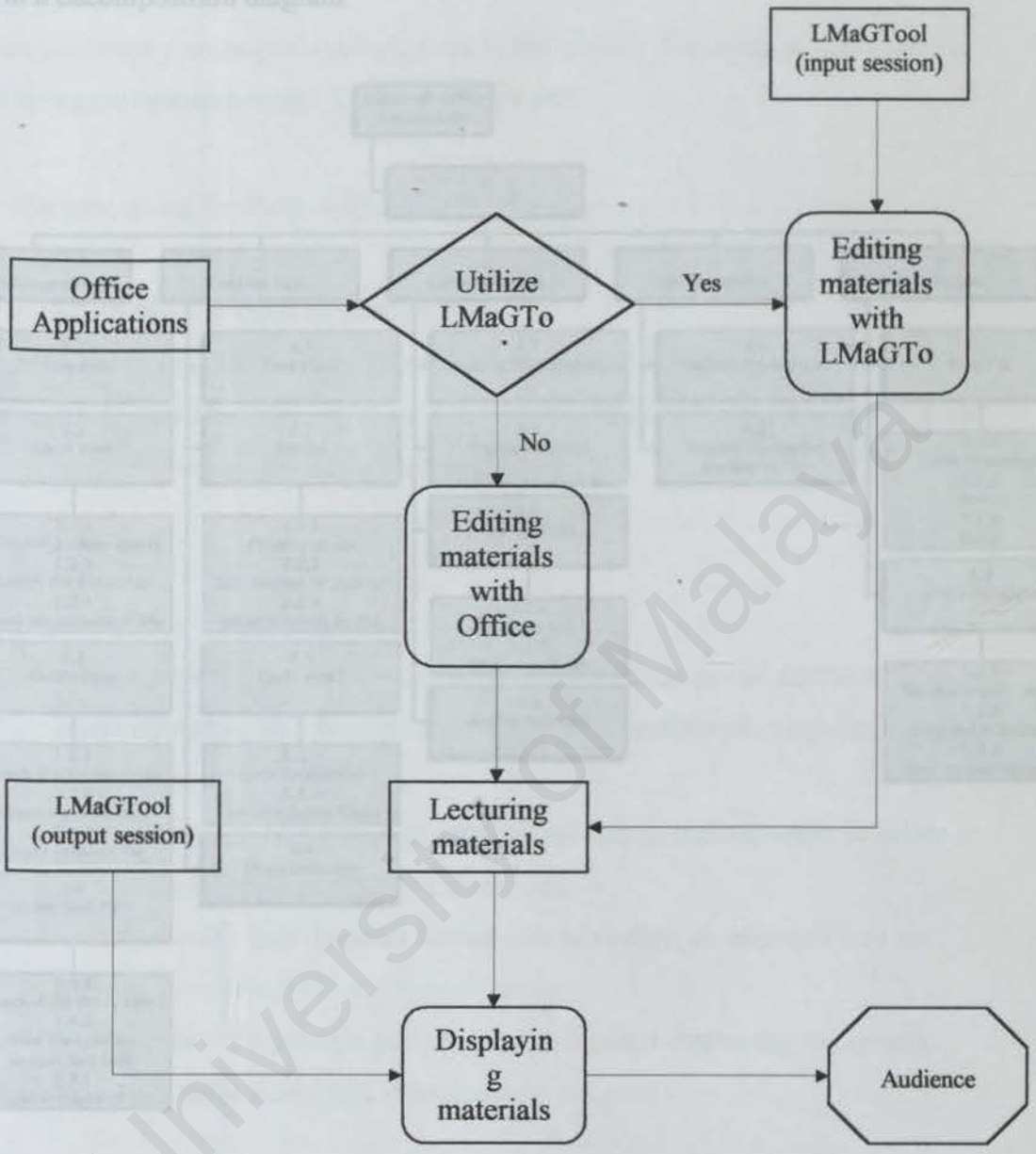

Figure 4.1: Project architecture flow diagram.

In Figure 4.2, more detail information about every functional task of the project is shown in a decomposition diagram.

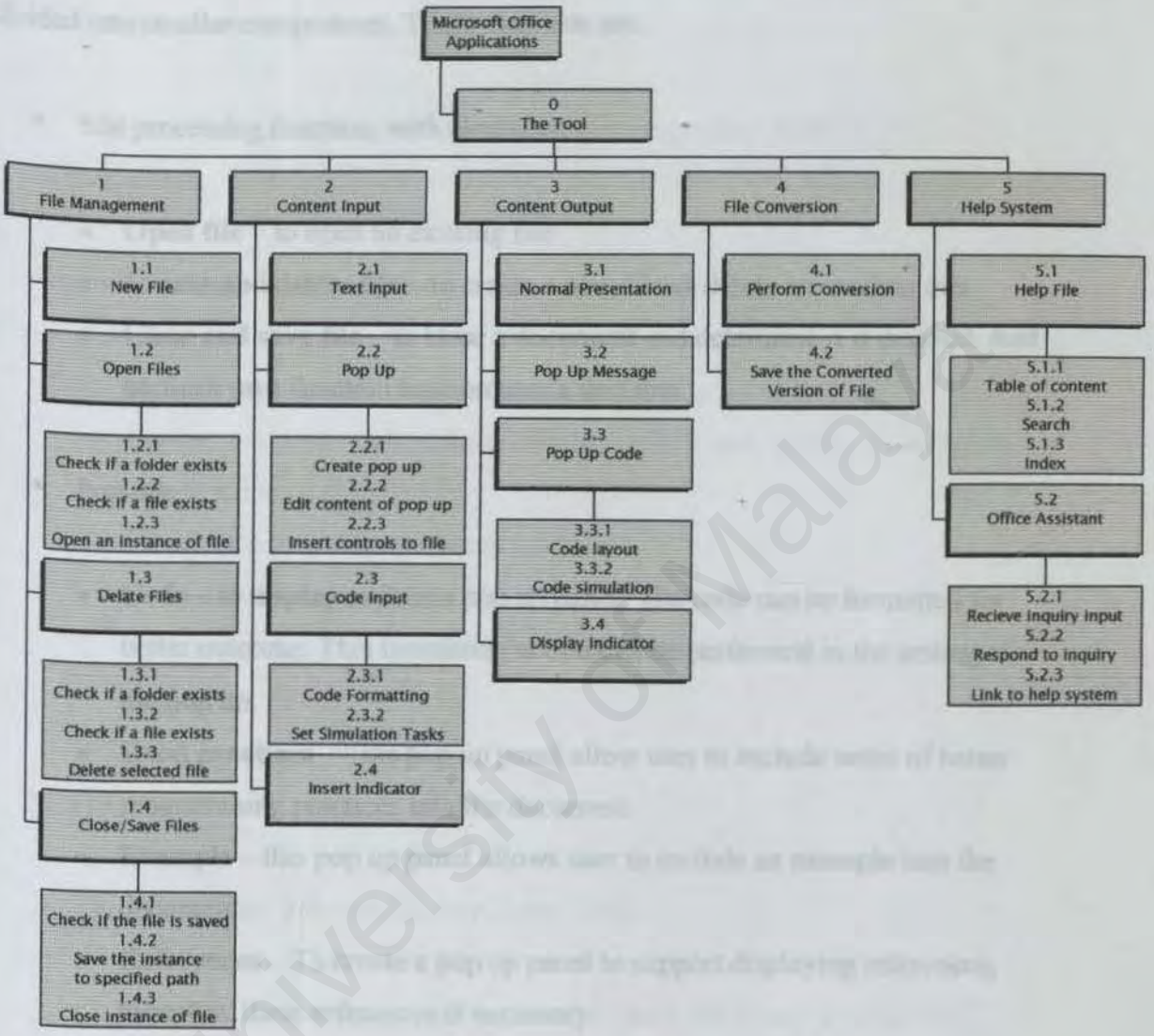

Figure 4.2: Decomposition diagram of the project.

4.2 Overview of project elements

There are potentially seven main components in the project. And each of them can be divided into smaller components. These elements are:

- File processing function, with elements:
  - × **Open file** – to open an existing file
  - × **Create and delete file** – to create a new file or delete an existing one
  - × **Close and save file** – to close a document and determine if it is saved. And perform save function base on user's decision.
- Pop-ups
  - × **Code** – to display code in a pop up panel. The code can be formatted for better outcome. This formatting action can be performed in the setting of the pop up.
  - × **Good practices** – This pop up panel allow user to include notes of better programming practices into the document.
  - × **Example** – this pop up panel allows user to include an example into the document.
  - × **References** – To create a pop up panel to support displaying references, hyperlink these references if necessary.
- Code simulation

  This element consists of functions to simulate code in way that user specified. The functions will simulate the outcome of the executing the code, such as
  - × **Messages** – Codes that displayed on screen to communicate with user of the code.
  - × **Input** – Display when the code executed to an input section. Value will be input by user.

- **Output** – Display results of execution of codes, such as results generated by processing input.

- Progress indicator

  To be include in a document to visualize the progress of viewing the document.

- User interface of the tool, include

  - **Menu items** – most common way to link user to functions
  - **Toolbar** – easier way to access functions of the tool, consist of clickable graphical bar.
  - **Graphics** – to decorate the user interface
  - **Panel** – to let user input setting or display messages to users
  - **Buttons** – basic user interface element. User click on a button to start an action.

- Help system

  The help system will provide help in two ways:

  - **Help files** – to be read by user who need more information of the tool
  - **Office assistant** – can assist user to solve problems step-by-step.

- Converting tool

  This element will help user to convert documents to other file format. Structured materials will be sorted into place during the file conversion. This helps students for a better understanding of the generated notes.

These elements are illustrated in Figure 4.3.

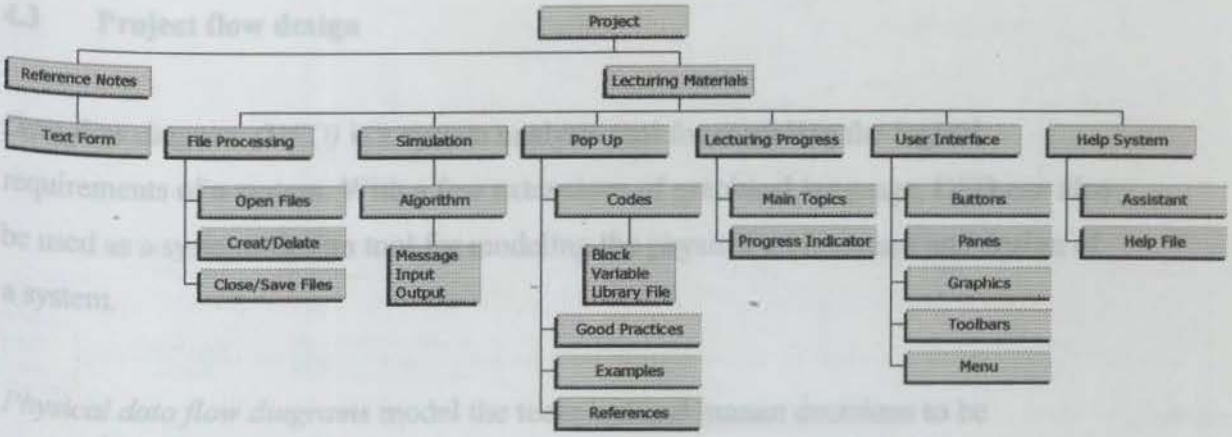

Figure 4.3 Basic element of the project

4.3 Project flow design

Data flow diagram (DFD) is a system analysis tool for modeling the logical requirements of a system. With a few extensions of graphical language, DFD can also be used as a systems design tool for modeling the physical architecture and design of a system.

Physical data flow diagrams model the technical and human decisions to be implemented as part of a system. They communicate technical choices and other design decisions to those who will actually construct an implement the system.

A physical data flow represents any of the following:

- The planned implementation of an input to or output from a physical process
- A database command or actions such as create, read, update, or delete
- The import of data from or the export of data to another system across a network, or
- The flow of data between two modules or subroutines within the same program

Based on the functional decomposition and structure of the project, processes in the project are grouped into 4 categories.

- File management
- Content editing
- Document conversion
- Help retrieving

These processes are illustrated in a data flow diagram as shown in Figure 4.4.

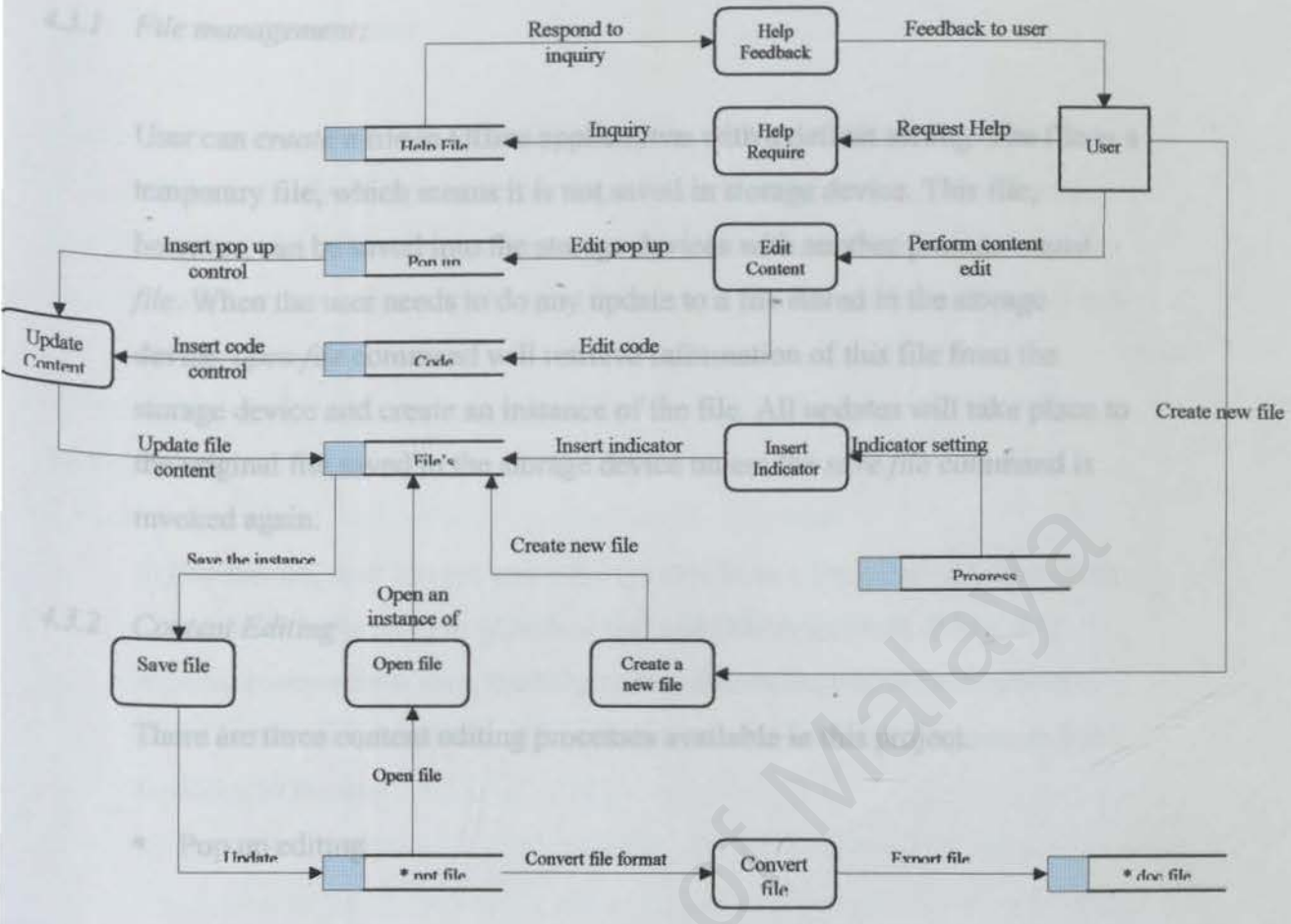

Figure 4.4: Project flow diagram

4.3.1 *File management:*

User can *create* a file in Office applications with a default setting. The file is a temporary file, which means it is not saved in storage device. This file, however, can be saved into the storage devices with another process – *save file*. When the user needs to do any update to a file stored in the storage device, *open file* command will retrieve information of this file from the storage device and create an instance of the file. All updates will take place to the original file saved in the storage device unless the *save file* command is invoked again.

4.3.2 *Content Editing*

There are three content editing processes available in this project.

- Pop up editing

  User can edit content of a pop up in this section. After editing process, a control to invoke the pop up will be placed into the document.

- Code editing

  User can edit content of code samples here and perform formatting to the code, such as highlight (colorize) the code and simulate setting of the code. After editing process, a control to invoke the code will be placed into the document.

- Insert indicator

  User can insert a progress bar to the document to indicate the progress of how far the document has been viewed.

4.3.3 *Document conversion*

In this section, user can perform conversion from the original file format to another specified format, such as from *Power Point* format to *Word* format. This function will rearrange the original document's content into order. After the conversion, a file with the specified format will be created and stored into storage device.

4.3.4 *Help retrieving*

In this section, user can get guide for the tool from a preinstalled help system. This system included help files, tool tips and Office assistant. When help requests receive from user, the help system determine which kind of help is useful to the user. After that, information about the tool will be generated and feedback to the user.

4.4 Input design

Most new applications being developed today include a graphical user interface (GUI). Most are based on Microsoft *Windows*. While GUI designs provide a more user-friendly interface, they also present more complex design issues than their predecessors.

Input design of the tool will focus on selecting the proper screen-based controls for entering data on a GUI screen. The design will implement some of the most common controls used in GUI-based input forms. These controls include:

- **Text box** – for entering single or multiple lines of text, such as notes or codes into the document. This control requires uses to type the data in the box.
- **Radio button** – Radio button provides the user with an easy way to quickly identify and select a particular value from a value set.
- **Check box** – A check box consists of a square box followed by a textual description of the input field for which the user is to provide the yes/no value. This control is used in the tool to let users select whether to on or off a function.
- **List box** – A list box is a control that requires the user to select a data item's value from a list of possible choices. List box is used in the tool for users to select their criteria to certain specifications, styles of display, and other edit selections.
- **Button** – Strictly speaking, buttons are not input controls. They do not contribute to the selection of or input of actual data. Nonetheless, input form design is incomplete without them. Buttons serve several purposes. They allow a user to commit all of the data to be processed, or cancel a transaction, or get help. They can be used to navigate between instances of the same form.

The tool consist of several input GUI screens, these screens are illustrated and described as below:

4.4.1 *Normal input*

This input screen is the input screen of Office applications; there is no modification done to the existing Office content input field.

4.4.2 *Pop up input screen*

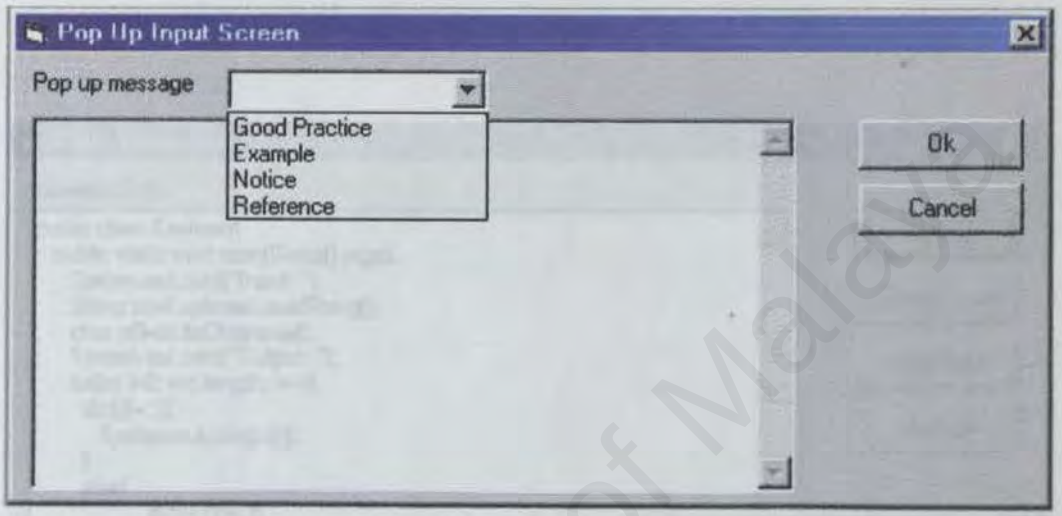

Figure 4.5: Input screen of Pop Up

The form will consist of:

- A drop down list for users to choose type of pop up they wish to insert into the document.
- A multiple lines text box for users to input their message.

4.4.3 *Code input screen*

The code input screen is a form consisting:

- A multiple lines text box for users to input their code
- Two check boxes to let user define whether to highlight the code and whether to allow code compare. If the *Highlight code* check box is checked, code will be highlighted as options specified in setting panel,

which accessible by clicking the *Setting* button. The setting is including which element of code selected to be highlight and colour for each element. If the *allow compare* check box is checked, users have to insert another code with the code input form and an addition control will be added to the code output control. All these actions can be processed when the *Format Code* button been clicked. After all formatting had been done; users can click the Insert Code button to insert their code/codes to the document.

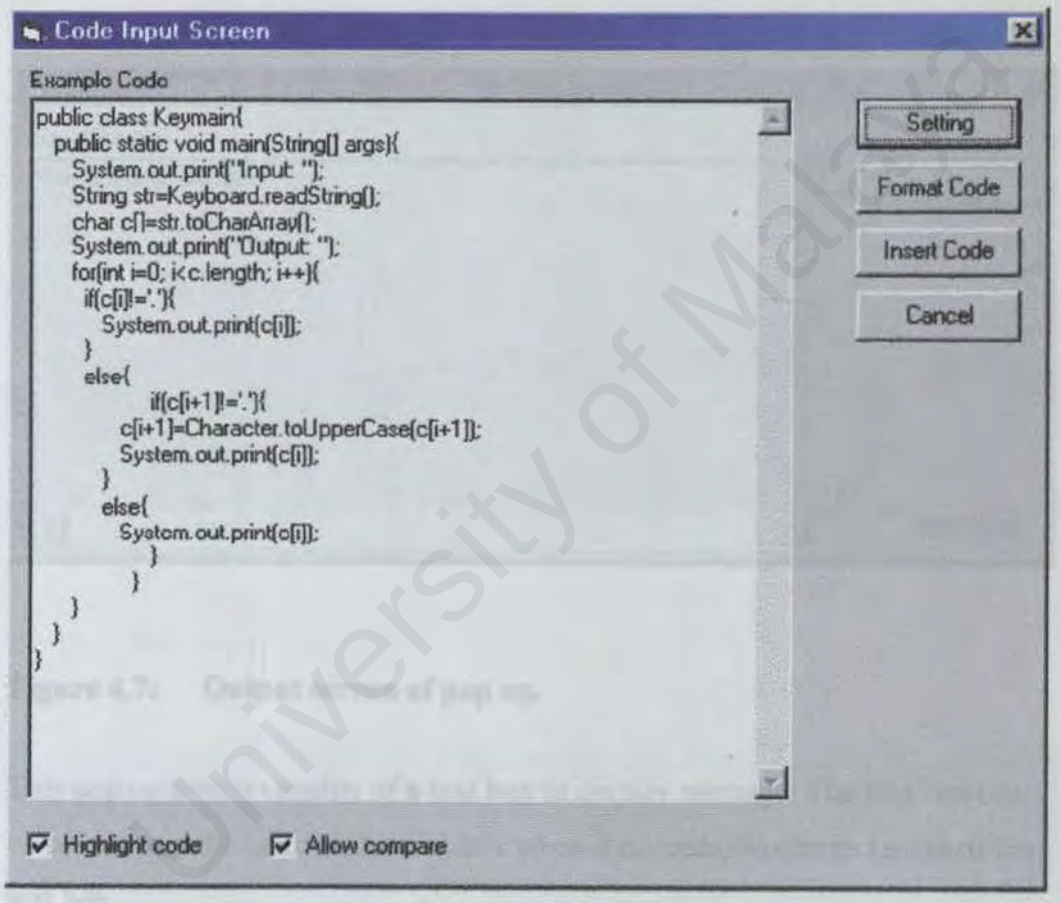

Figure 4.6: Code input screen.

4.4.4 *Insert indicator setting panel*

In this panel, users can select styles of indicator, location on the screen to insert the indicator and other attributes of the indicator.

4.5 Output design

Output design for the tool implements same elements for GUI as in input design. The purpose of output design for this tool is to display users input information. This information, such as formatted code and progress indicator cannot be displayed by normal Office applications' functions. Following are some basic design of the output screens.

4.5.1 Pop up output screen

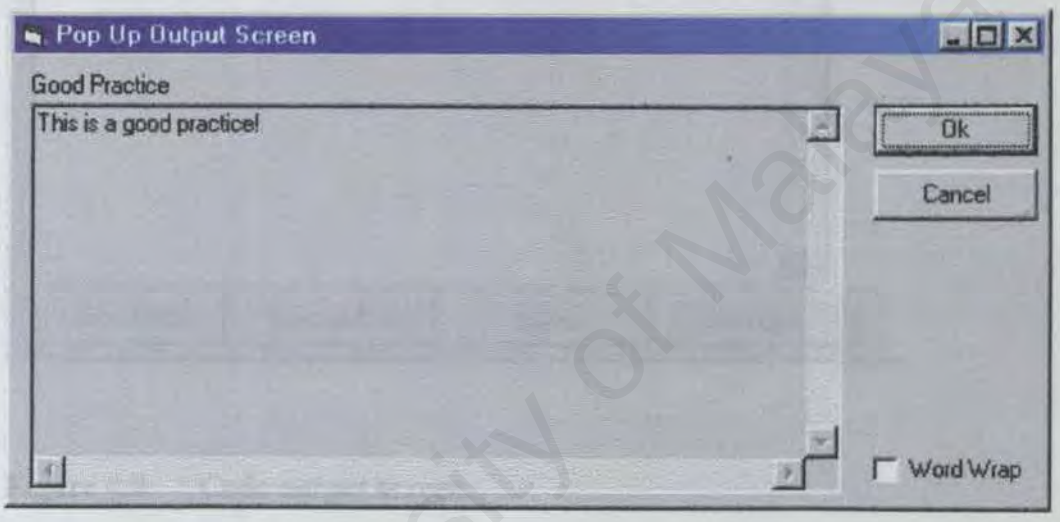

Figure 4.7: Output screen of pop up.

This pop up screen consists of a text box to display message. The text box can be set to wrap the word in the text box when it exceeds maximum length of the text box.

4.5.2 Code output screen

Code output screen consist of one or two text box, depends on whether there are codes to be compared. When there are codes to compare, user can click the *Compare Code* button to bring up another text box to show another set of code. User can set attribute to display these codes, such as arrange these codes into horizontal or vertical position. When user clicks on the *Show Block*

button, the tool will show each block of code separately to make the code more readable. *Run code* button simply invoke simulation of the code.

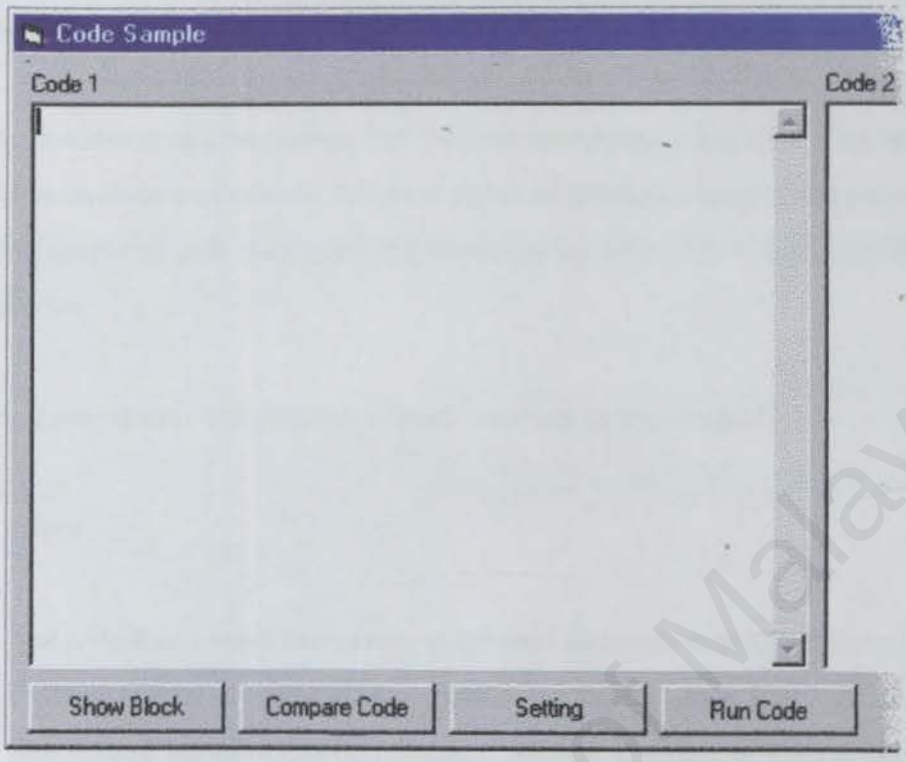

Figure 4.8: Code output screen.

4.5.3 *Progress bar output*

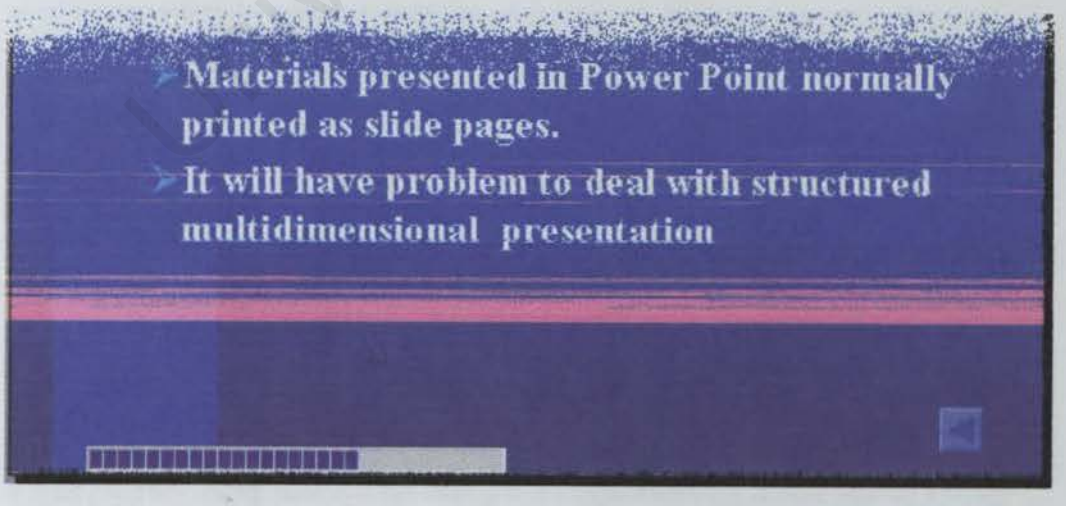

Figure 4.9: Progress bar been placed at the bottom of a slide.

The progress bar output is simple. A progress bar is place on a location of the document as specified earlier. And this progress bar just takes a little place in every page of document.

4.6 User Interface Design

User interface is the specification of a dialogue or conversation between the system user and the computer. This dialogue generally results in data input and information output. There are several styles of *graphical user interfaces (GUI)*. Traditionally these styles were viewed as alternatives, but they are increasingly blended. This section presents an overview of several different styles or strategies used in the project for designing graphical user interfaces and how they are incorporate with existing Office user interface.

Following are several components of user interface in this project:

4.6.1 Menu

The oldest and most commonly employed dialogue strategy is menu selection. Different types of menu cater to notice and expert users. In this project, the tool related menu would be put under one menu group, which reside at the existing menu bar of Office applications. As illustrated in Figure 4.10, the menu group "LMaGTool" (short for Learning Materials Generating Tool) is resides at the end of the menu bar. Users can easily access functions in the tool by select an action from the menu.

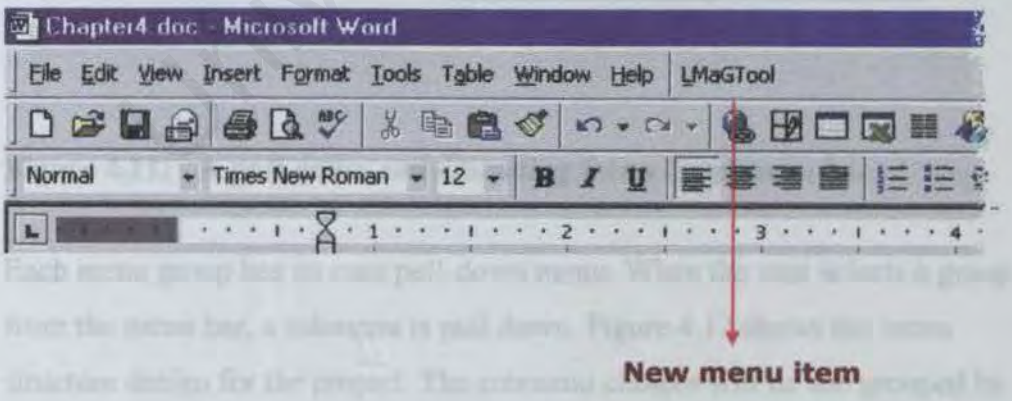

Figure 4.10: New menu item integrated into existing menu bar in Office

In a GUI, menus are usually implement with pull-down and cascading menus from a menu bar as shown in Figure 4.11. Users can select a menu group either using the mouse or a keyboard shortcut (e.g., simultaneously pressing the Alt-key plus the underlined letter of the menu, called a *mnemonic, shortcut,* or *hot-key*).

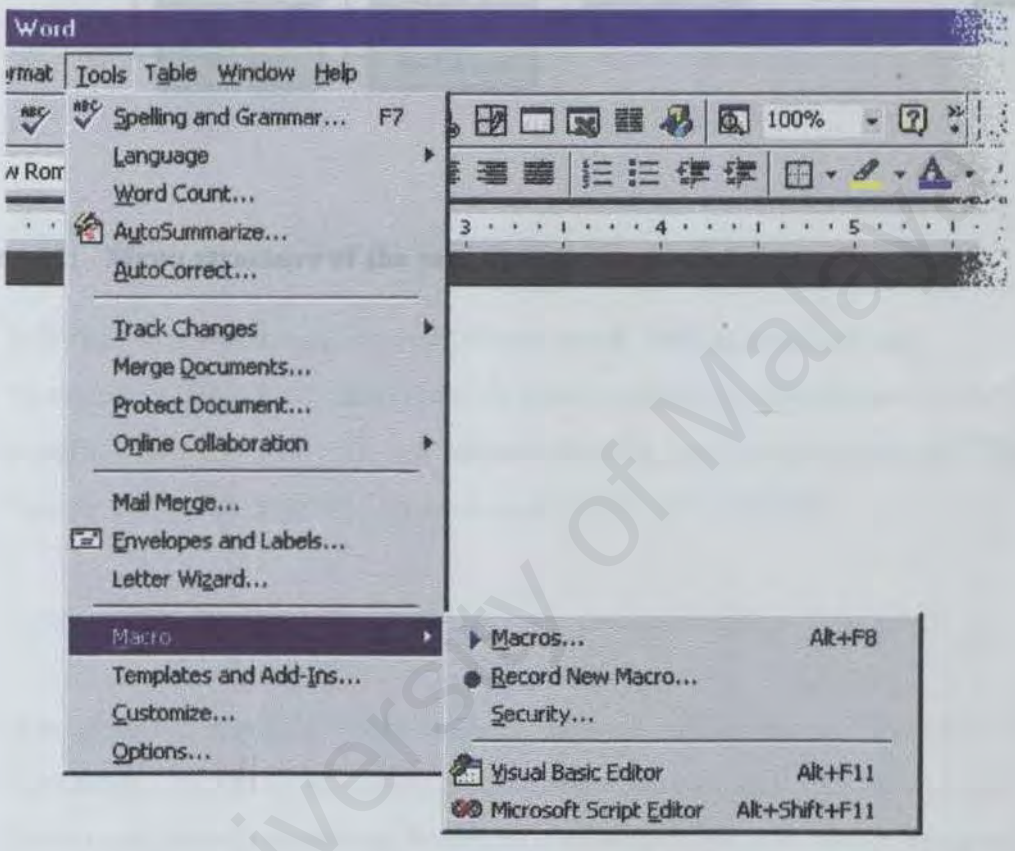

Figure 4.11: A pull-down and cascading menu from menu bar.

Each menu group has its own pull-down menu. When the user selects a group from the menu bar, a submenu is pull down. Figure 4.12 shows the menu structure design for the project. The submenu choices will be sub grouped by horizontal lines (e.g., grouping all *Save* or *Print* submenu commands).

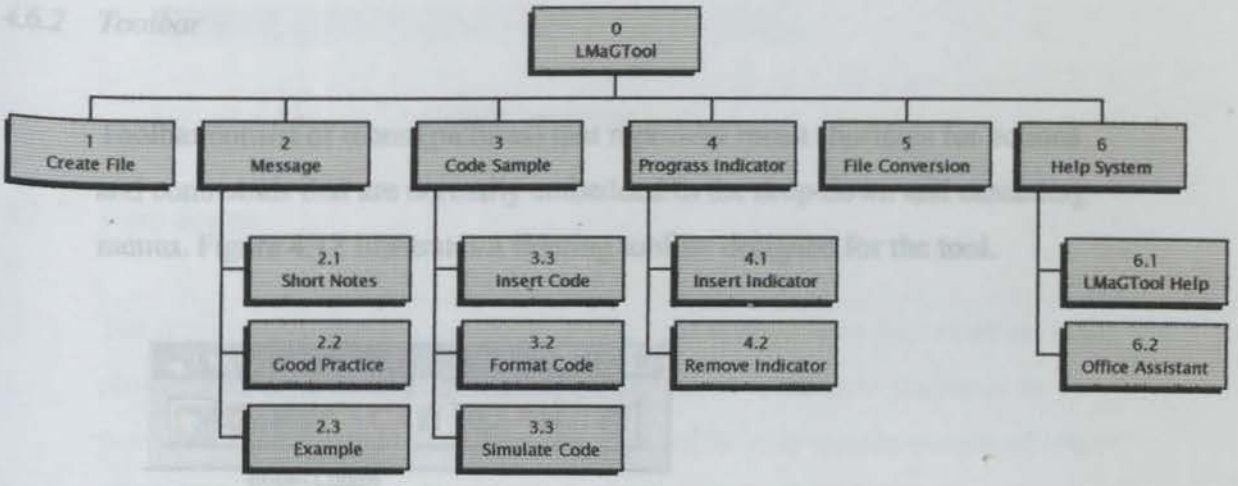

Figure 4.12: Menu structure of the project

4.6.2 *Toolbar*

Toolbar consist of icons (pictures) that represent menu shortcuts for actions and commands that are normally embedded in the drop-down and cascading menus. Figure 4.12 illustrates a floating toolbar designed for the tool.

Figure 4.13: A floating toolbar designed for this project.

In windows applications, a toolbar of commonly used actions is found immediately beneath the menu bar. The user can click on any of these tools or icons to immediately invoke that action without going through the menus. This toolbar will is customizable by users under Office environment.

4.6.3 *Progress bar*

A progress bar is a feature introduced to improve lecture timing. This little bar can shows how many percent of learning materials had been viewed; and how many more materials to show. With this feature, lecturer can control the speed and time to spend on each part of the materials. If lecture time is running out but there is a lot of materials have not been explain, lecturer may be cut short some less important part of materials. And if there is plenty of time, lecturer can spend more time explaining important focuses of the study.

The progress bar can avoid scrolling up and down to check how much materials left to present to the student, which disturb attention of students. On other hand, students can be more aware of the progress of the lecture process. Therefore they can easily focus on each part of the lecture, not wondering if the lecture is going to last very soon. Figure 4.13 illustrates a progress bar.

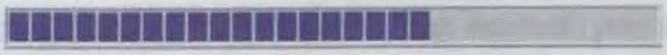

Figure 4.14: A simple progress bar

4.7 Help design

The design, construction, and testing of a help system is an important part of a project. Help system is expected to assist users to solve their problems by providing information about the tool. A complete help system consist of table of contents, numerous instructions, examples, and a thorough index.

Another help feature is more effective for the more novice user. This feature is help agent (or assistant). Agents are reusable software objects that can operate across different software applications and even across network. As is illustrated in Figure 4.14, a help agent is consisting of several elements.

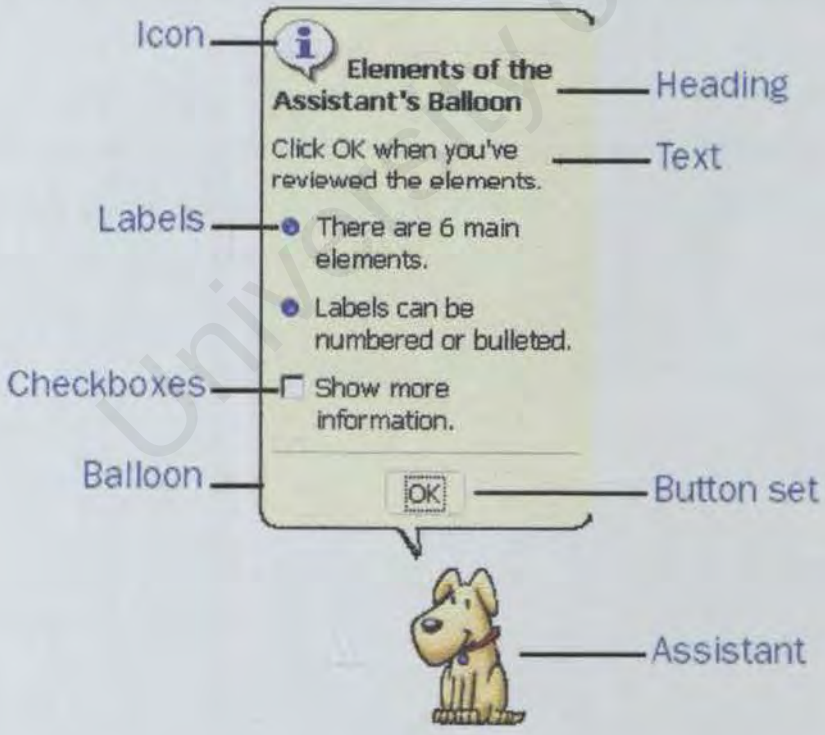

Figure 4.15: Elements of the Assistant's Balloon

Microsoft help agent (referred to as an assistant) provides a common help assistant across all Office 97 and Office 2000 applications. A single user click on this help agent will initiates help.

The Microsoft help agent is complemented by natural language processing technology that allow the user to write an inquiry in natural language phrases that are interpreted by the agent to present the most likely help responses. The user can then select one of those responses or enter into the more detailed help index.

Chapter 5

System Implementation

Chapter 5 System Implementation

5.1 Introduction

This chapter describes system implementation of Learning Material Generating Tools (LMaGTools). System implementation process included all coding approach, style of coding, and programming language issues.

As planned and decided in section 3.3 – Consideration of Development Tool, the project has been carried out with Visual Basic as the primary programming language to code, implement and test functions in LMaGTools.

Besides Visual Basic, a simple yet powerful tool provided by Microsoft Visual Studio, Help Workshop, had been used to edit, compile and test User Guide for LMaGTools. This tool is able to compile the user guide into common windows help file format, which is a 32-bit help file.

A own customized Rich Textbox, which has a lot of different and value added functions compares to existing Rich Textbox provided in Visual Basic reference, is imported into the project development environment. This Rich Textbox has been use to produce many text effects in "Code Highlight" section.

Following part of this chapter is detail of these processes and steps in the project lifecycle.

5.2 System Coding

System implementation is a process of transforming the design specification and data models into an executable software. As mentioned in the previous chapters, the programming tool that used to develop LMaGTools is Visual Basic and it is implemented in Office PowerPoint environment. In order to use the ADE of Visual Basic, a good understanding of the way it works and some new terminology and programming concepts associated with it is necessary. These concepts include COM, ActiveX, OCX, API, etc. In addition, there are a few other aspects that have to be considered, such as simplicity of code and run-time efficiency of the written program.

5.2.1 Visual Basic

Visual Basic is a simple yet powerful language that supports Object Oriented Programming (OOP). With its compact IDE, a system can be built in fastest and most cost effective way. More important is Visual Basic provides a direct access to the common Office objects, including objects in PowerPoint. To access these object and manipulate it for own use, a reference to Office can be made by setting the reference of the project (system) property.

Visual Basic provides a number of outcomes of system for system developers to choose from. These outcomes including standard EXE, ActiveX EXE, ActiveX DLL and other project type, the dialog to let users choose their project type is shown in Figure 5.1.

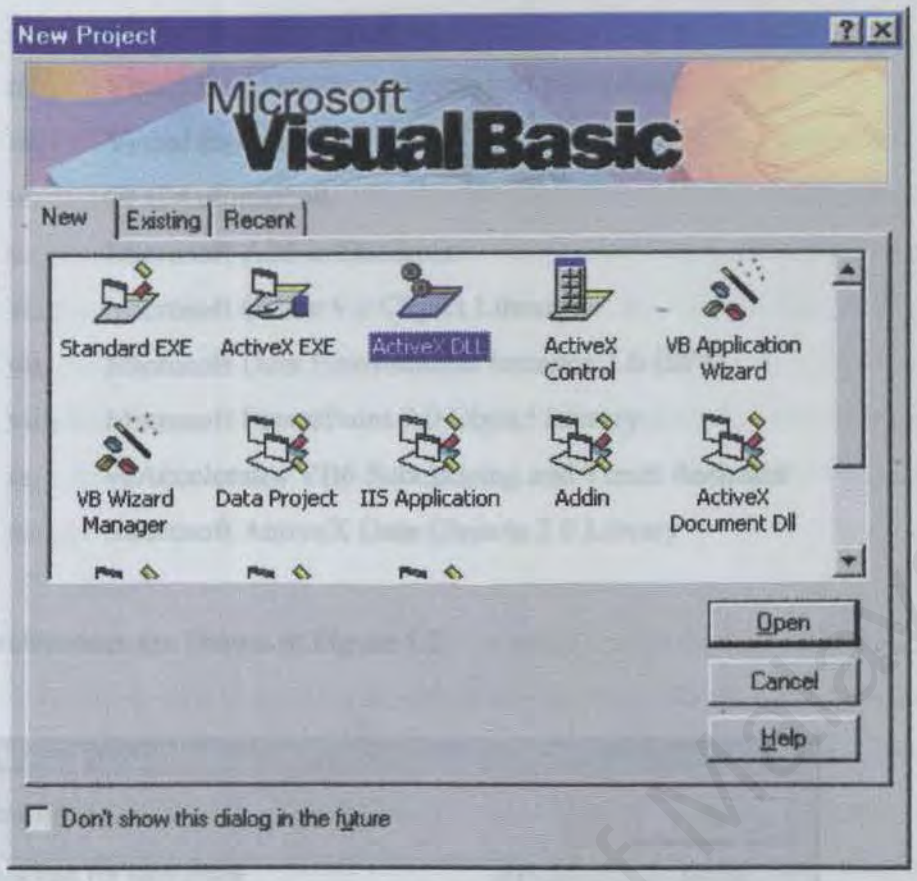

Figure 5.1: Dialog for user to choose their project type

From these various project type, ActiveX DLL is chosen to be LMaGTools' project type as this is among best ways to plug add-in features to Office applications.

ActiveX DLL project will always generate application extension file (*.dll) as the final outcome. The application extension file is then able to act as a library file for Office or other application. In this project, the final outcome will be connected to PowerPoint.

5.2.2 *Reference for the Project*

To gain access to certain functional element, a few references would be chosen as reference for the project. There are total of ten references made in this project, including:

i. Visual Basic for Application

ii. Visual Basic runtime objects and procedures

iii. Visual Basic objects and procedures

iv. OLE Automation

v. Microsoft Add-in Designer

vi. Microsoft Office 9.0 Object Library

vii. Microsoft Data Environment Instance 1.0 (SP4)

viii. Microsoft PowerPoint 9.0 Object Library

ix. vbAccelerator VB6 Subclassing and Timer Assistant

x. Microsoft ActiveX Data Objects 2.0 Library

These references are shown in Figure 5.2

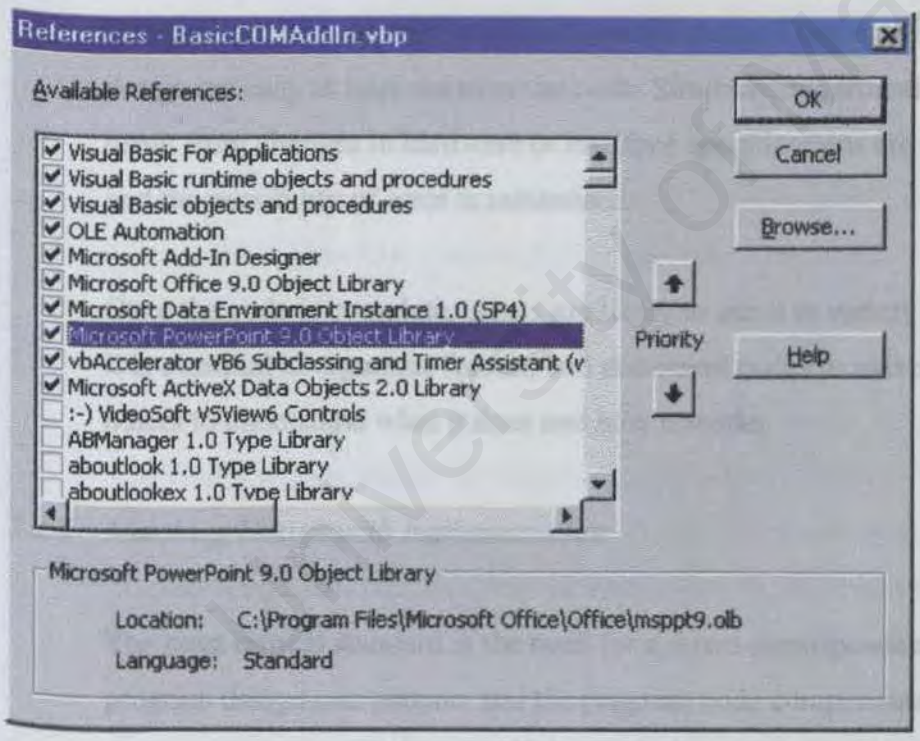

Figure 5.2: Reference for the project

5.2.3 *Programming Standards*

There are a few standards and procedures to follow during coding process take place in system development lifecycle. Standards and procedures can helps to organize our thoughts and avoid mistakes. One of the procedures involves methods of documenting the code so it is clear and easy to follow. Such documentation allows us to leave and return to our work without losing track of what we have been doing. Standardized documentation also helps in locating faults and making changes, because it clarifies which sections of the program perform which functions.

Standards and procedures also help in translating design to code. By structuring code according to standards, we maintain the correspondence between design components and code components. Consequently, changes in design are easy to implement in the code. Similarly, modifications to code that result from changes in hardware or interface specifications are straightforward, and the possibility of error is minimized.

Once the code is complete, others are likely to use it in variety of ways. Thus, it is essential to organize, format, and document codes to make it easy for others to understand what it does and how it works.

5.2.4 *Matching Design with Implementation*

The most critical standard is the need for a direct correspondence between the program design components and the program code components. The entire design process is of little value if the design's modularity is not carried forward into the code.

5.3 Project Programming

Project design is a guide to the function and purpose of each component. Program components of this project involve three aspects: control structures, algorithms, and data structures.

5.3.1 Control Structures

Many of the control structures for a component are suggested by the architecture and design, and we want to preserve them as the design is translated to code. We know that modularity was a good design attribute. By building a program from modular blocks, we can hide implementation details at different levels, making the entire system easier to understand, test and maintain. We can consider a program component itself to be modular, and we use procedures, subroutines, methods, and inheritance to hide details while enhancing understandability. Moreover, the more modular the code component, the more easy it can be maintained and reused; modification can be isolated to a particular subroutine or other subcomponent.

5.3.2 Algorithms

The program design often specifies a class of algorithms to be used in coding the component. We have a great deal of flexibility in converting the algorithm to code, subject to the constraints of the language (Visual Basic) and hardware.

One of the area we have great discretion is the performance or efficiency of the implementation. We tent to code the way that makes the code execute as fast as possible. However, we have to pay the cost for faster code:

i. The faster code is more complex and thus take more time to write

ii. The cost of time to test the code, whose complexity requires more test cases or test data

iii. The cost of time for others to understand the code

iv. The cost of time to modify the code, if necessary

Before taking any approach, we must balance execution time considerations with design quality, standards and system requirements.

5.3.3 *Data Structures*

In writing program, we should format and store data so that data management and manipulation are straightforward. Several techniques are used to organize the program.

i. Keeping the program simple.

The program's design may specify some of the data structures to be used in implementing functions.

ii. Using data structure to determine a program structure.

Program structure is depends on whether a recursive data structure or non-recursive data structure is involves. If a data structure is recursive, then a recursive program structure would better fixed to data structure.

iii. Localizing input and output

Those parts of a program that read input or generates output are highly specialized and must reflect characteristics of the underlying hardware and software. Because of this dependence, the program sections performing input and output functions are sometimes difficult to test. In fact, they may be the sections most likely to change if the hardware or software if modified. Therefore, is desirable to localize these sections in components separate from the rest of the code.

5.4 Summary

System implementation is a most important part of the whole developing lifecycle, as it will directly affect the reliability, usability and readability of the final program. In this chapter, we had made a clear system implementation approach. This approach including choosing a suitable development tool, Visual Basic. Besides that, three important element in programming: Control Structures, Algorithms, and Data Structures are implemented carefully to ensure maximize the performance and minimize the risk of program failure.

In next chapter, System Testing, we will test the system from small pieces of modules to the integration of whole system. The way to conduct system test is complicated and must be well handle to ensure the system is free of error. We will describe ways of conducting test, ways to solve problems, and so on. Buttons and functionality of components will also be tested to verify the accuracy of outcome.

Chapter 6

System Testing

Chapter 6 System Testing

6.1 Introduction

This chapter describes modules and methods used for system testing. Main purpose of testing the system is to make sure that the entire system does not perform any illegal operation, and does not consume too much system resources.

The functionality of user interface elements such as buttons, text boxes, menus and combo boxes has to be test for correctness of link of functions, usability, and efficiency. System flow for every individual part of the system has to be check for optimization of executions to produce a reliable system.

System output has to be test for its correctness. If output of the system is differ from what is expected, then it indicate that some error may occur in some part of the system code, or even the whole concept of system flow. These output are tested from time to time to avoid accumulating mistake or error in information storing, retrieving, analyzing, formatting and printing.

Process of developing LMaGTools involves a series of production activities opportunities for injection of fallibility is enormous. Errors may begin to occur during the very inception of the process, where the objective maybe erroneously imperfectly specified, as well as errors that occur in later design and development stages. Hence, system testing is a critical process in the whole system development lifecycle. It is important in software quality assurance and represents the ultimate reviews of specification, design and coding.

If testing is conducted successfully, most errors in the software will be discovered in the early stage. As another benefit of conducting system testing, it demonstrates that the software functions appear to be working as what it supposes to be.

6.2 Type of Testing

Glen Myers [MYE79] states a numbers of rules that can serve well as guidelines for testing LMaGTools:

- Testing is a process of executing a program with that intent of finding an error
- A good testing case is one that has high probability of finding an as yet undiscovered error.
- A successful test is one that uncovers an as yet undiscovered error.

The testing process is partitioned into several phases:

- Objective and Specification Review
- Software Testing
    - Unit Testing
    - Integration Testing
    - System Testing

6.2.1 *Objective and Specification Review*

In order to verify that the system has been developed in accordance with the initial scope and specifications, several reviews have been held from time to time. A checklist of scopes and specifications to be fulfill is made and refine actions was done on the system when contrast with the scopes or specifications.

6.2.2 *Unit Testing*

During performing unit testing to LMaGTools, verification effort is focused on the smallest unit, sub functions. Using detailed description as a guide, important control paths and variable values parsing are tested for possible error. White Box Technique is applied during this test, which means involving inspection of internal program flow of every module. The sequence of the unit testing is:

i. Test data across a module interface

ii. Test the local data structure

iii. Test the boundary conditions

iv. Test every independent execution path of the module

v. Test the error handling paths

6.2.3 *Integration Testing*

Even though modules work individually, they may not work when we relate them to each other. Data can be lost or wrongly define across modules; one module can have an inadvertent, adverse effect on another sub functions, when combined may not produce desired final outcome; global data structure for certain cross module data can present problems. Hence, integrating testing is done on every combination of modules, which work together until all possible error is found and fixed.

Black Box Technique is applied to conduct the test, which means every module is treated as a black box. Only the external behavior and the interface of modules need to be considered. The test is conducted by using Incremental Integration Approach. Following this principle, individual module is added to the system one by one. This eases the isolation of error sources.

6.2.4 *System Testing*

Lastly, LMaGTools has undergone system testing to verify that all system elements have been properly integrated and perform allocated functions. The testing procedure includes the following test:

i. Recovery Testing

Te purpose of this test is to verify that recovery is properly performed when LMaGTools counters system failure.

ii. Security Testing

This test attempts to verify that security mechanisms build into LMaGTools will, in fact, protect it from improper penetration.

iii. Stress Testing

The main objective of this test is to confront LMaGTools with abnormal situation including low memory and low system resources.

iv. Performance Testing

Performance testing is designed to test the run-time performance of LMaGTools. It assures the length of processing time of LMaGTools is falling into acceptable range.

6.2.5 *Execution of Testing and the Solution*

The following is the testing executed to LMaGTools:

i. Button Testing

There are many buttons been created and used in LMaGTools. As an example in Figure 6.1, when the <Save> button is clicked (1), the program will look for value of active code panel selection in ratio button group located at the control panel (2). As for this example, "code 2" is set to be active code panel, therefore formatted text in code panel 2 (3) will be save by the "save" function.

This happens to be same for other buttons such as <New> button. When the <New> button is clicked, active code panel will be reset to a clean text box. But to ensure users not accidentally lost all their work if they wrongly clicked the new button, a verification message box will be pop up requiring users decision to save their work.

All buttons, ratio buttons, combo boxes and other common user interface elements had been tested and all of these elements response correctly to the users' actions.

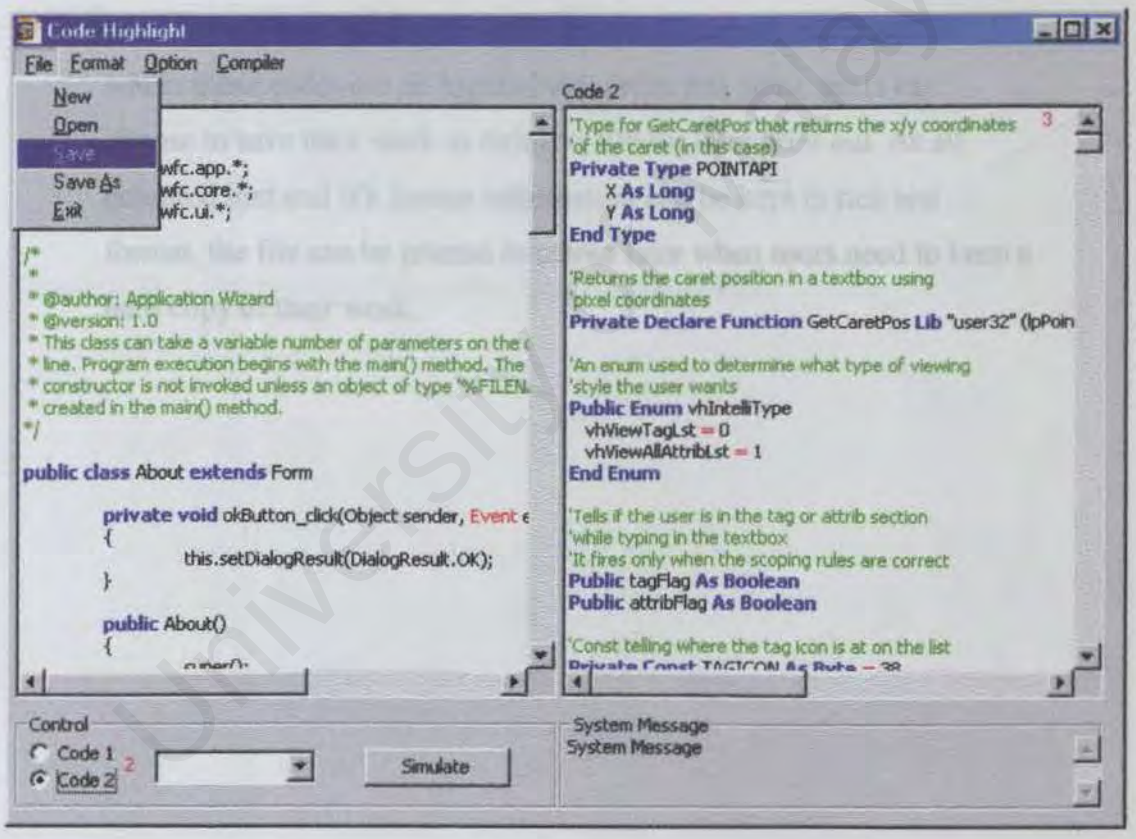

Figure 6.1: Every button is tested for their functions

ii. Function Testing

There are many functions in LMaGTools such as Code Highlight, Code Compare, Insert Message, and Convert File Format. As an example in Figure 6.2, users can type and edit their programming code in the code section. All text type in this while is black in colour. When users set the syntax they want to relate to the code and perform highlight to the code (via format menu), text in code section will be coloured and formatted according to the highlight function. This function will refer to external syntax file for recognizing every syntax of the code and format it accordingly. The next step will be displaying the result of highlighting to users. This is illustrated in Figure 6.3, where those codes are all highlighted. From this point, users can choose to save their work as rich text file for later print out. As all coloured text and it's format information can be kept in rich test format, the file can be printed in colour later when users need to keep a hard copy of their work.

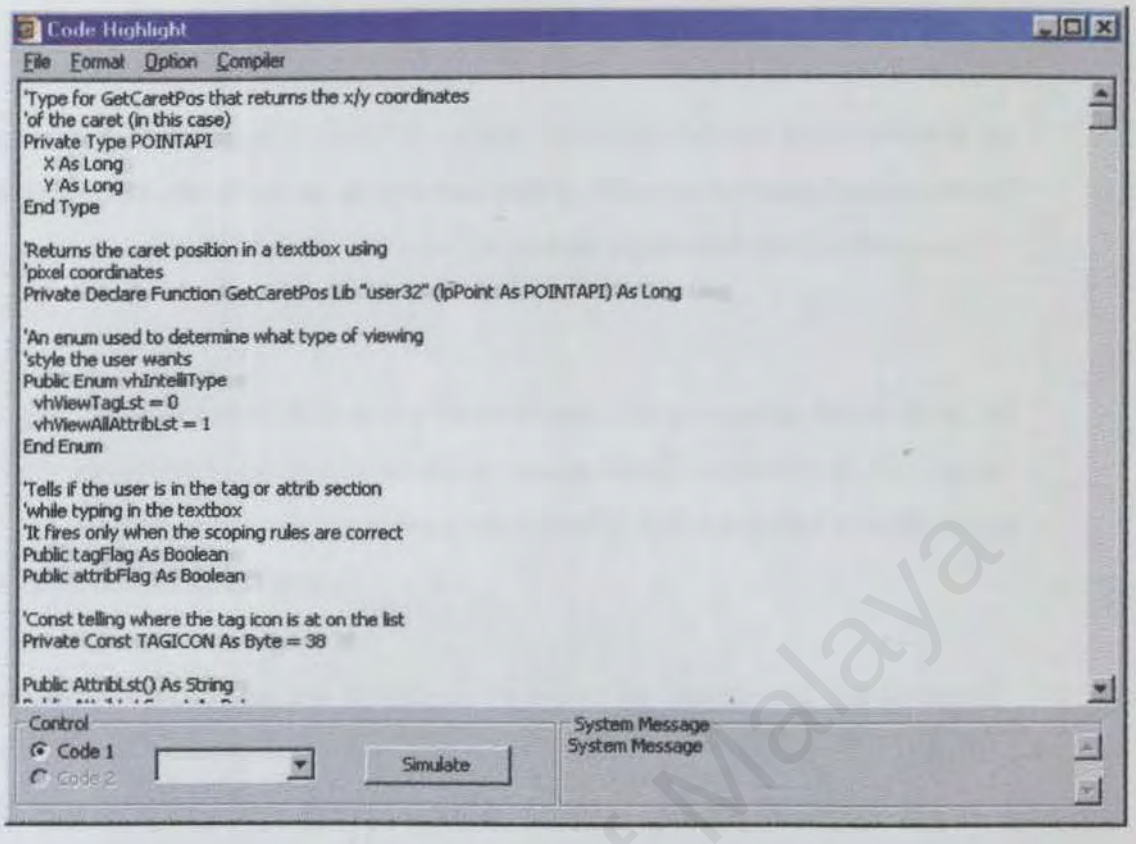

Figure 6.2: Users input plain code in the code section.

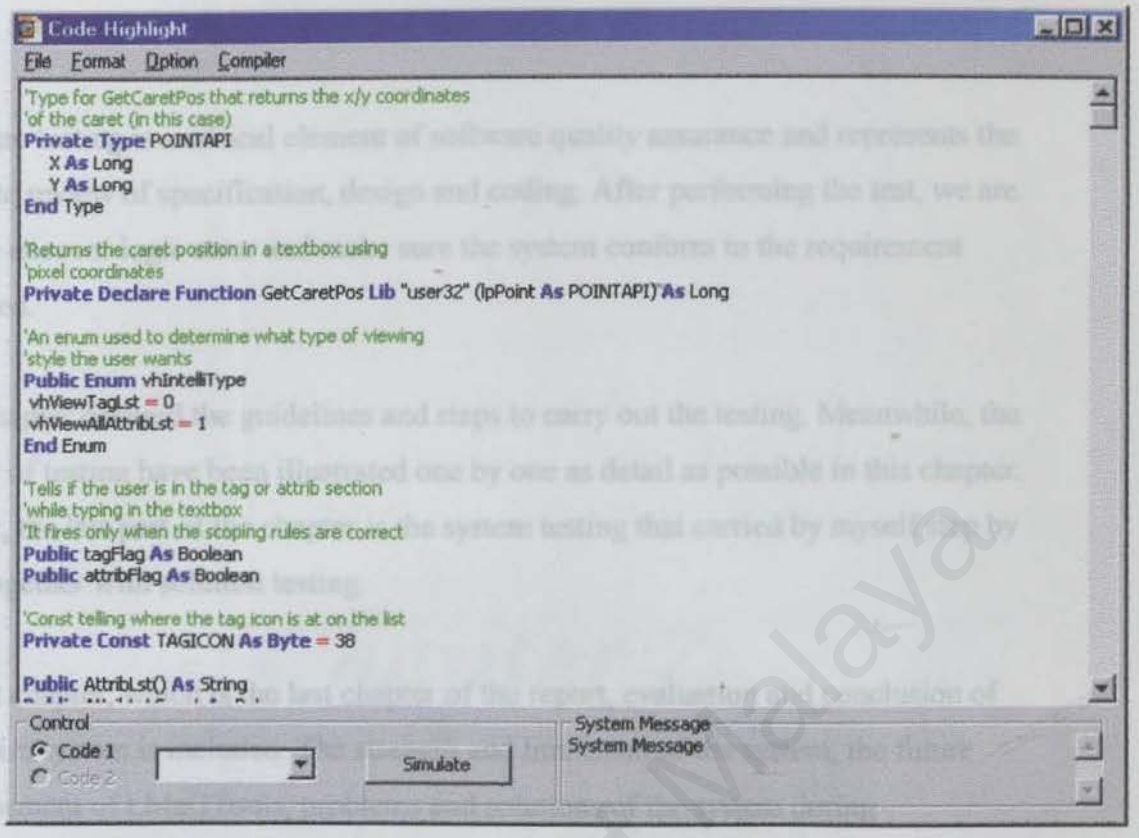

Figure 6.3: User use highlight function to highlight the code

iii. Output Testing

There are two type of output in LMaGTools – The on screen display output (including code window and slide window) and storing information in hard drive. As shown in Figure 6.3, highlighted code is displayed correctly in the code window.

6.3 Summary

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. After performing the test, we are able to uncover logic error and make sure the system conform to the requirement specified.

This chapter covered the guidelines and steps to carry out the testing. Meanwhile, the variety of testing have been illustrated one by one as detail as possible in this chapter. Finally, the last part of the chapter is the system testing that carried by myself step by step, together with solution testing.

In next chapter, which is the last chapter of the report, evaluation and conclusion of the entire system is included. The strength and limitation of the system, the future enhancement of LMaGTools, problems and solutions of the system during development period, and also knowledge and experience gained from the system development will be covered.

Chapter 7
System Evaluation

Chapter 7 System Evaluation

7.1 Introduction

This chapter is the final chapter that concludes the overall solution of this project. At the beginning of this chapter is the illustration of problems encountered when developing LMaGTools and solutions to solve the problems. Meanwhile, the ways to evaluate the system has been described with detail by interviewing end users.

Besides, the system strengths and system constraints have been listed out as detail as possible in this chapter. This is important so that every weakness and loophole in the system can be improved in the future enhancements. By doing this project, I have been trained and learn a lot of knowledge and also gain real life programming experience, which is much different from learning in lecture hall.

As I develop the system, I have exchange a lot of idea and experience with experience programmers, classmates, friends and also peoples who willing to help from the other end of the world, via Internet.

Finally, I have made an overall conclusion for this project and this is a simplify explanation of what I have done in this project.

7.2 Problems and Solutions

I have come across several problems during the process of developing LMaGTools. Including:

i. *Difficulty in communicating with interviewees during survey*

Since computer-based learning materials generating is rather a new idea, many of the interviewees could not grasp its concept properly, particularly those who is not computer literate. They tend to be confused by the computer technical terms.

Solution:
I'd tried to improve to situation by explaining to them in plain language.

ii. *Difficulty in determining an appropriate user interface standard*

The functions of the system are rather complex. After all, I have to make sure user interface is as simple as possible for the sake of user.

Solution:
Referring to the *Graphical User Interface Convention, Guides, And Tips*, publiched by Microsoft Press solved large portion of this problem. Feedback from users was also gathered from users from time to time.

iii. *Lack of mastery in some advance programming skills*

In the code highlight section of the system, many functions and coding need a high skill in advance programming. To move from hard coding to more powerful and efficient syntax referencing technique require experience in exploiting the Language, Visual Basic.

Solution:

I had asked an experience programmer for his experience and try to observe his skill in solving similar problem. Besides, I also post many problems I face in the online discussion forum for programming language. I get a very impressive feedback from the other end of the computer.

iv. *Difficulty of Implementing MDI form in ActiveX DLL project*

I planned to implement MDI forms in code comparing feature, but MDI form is not an option in ActiveX DLL project.

Solution:

Use two Rich Text Box, hiding one of them, and determine their size by monitoring the form's resizing event. This is not an ultimate solution since users cannot open more then two codes for compare. But it still reduce the impact of the problem in the sense that not many users would compare three or more codes at the same time, especially when teaching programming language.

v. *Difficulty in evaluating the system*

I have face many problems in finding the system constraint and system evaluation. I cannot perform a real balance evaluation because the system is build by myself.

Solution:
Rather than making an imbalance evaluation, I had conducted an interview and demonstrate the system to a group of end users. From this interview, many facts that never cross into my mind are appeal by the end users.

vi. *Hardware failure problem*

During the whole system development progress, especially the starting phase of the project, I face problem of hardware failures, which force me to reformat the hard drive, and leads to lose of important research information and result. This failure had also slow my developing steps for I have to replace a whole complete operating system and needed developing software such as Microsoft Office and Microsoft Visual Studio.

Solution:
I backup my research progress log and result more often and in multiple media to prevent lost of data and information. Besides that, a hard disk mapping software, Norton Ghost, is installed and implemented to reduce time needed to reinstall everything when the system crash, it is very efficient since I'd only spend fifteen minutes to reset my computer in the second crash instead of hours in my first system crash.

7.3 Evaluating by End User

To evaluate the overall system for LMaGTools, an interview has been conducted to find out the strength and limitation of the system. Several categories of possible end users from different field of study have been chosen to be the interviewees. Most of them are student, who study Information Technology and Engineering in University of Malaya. Others are people posses with working experience, especially in computer related field.

Firstly, I demonstrated the system and explain the system functions as detail as possible to them. Then, they were asked to fill a survey form, which I prepared earlier (as shown in Figure 7.1). An explanation on the question was done when they get confused with the questions.

Questionnaire – Testing on End User
Learning Materials Generating Tools (LMaGTools)

Name : _____
Age : _____
Field of Study : _____
Occupation : _____

Please relate the question with the system demonstrated to you.

1. Do you think the user interface is attractive and user friendly?

   _____
   _____
   Other Comment: _____
   _____

2. Is the system easy to use?

   _____
   _____
   Other Comment: _____
   _____

3. Are the functions of the system enough for the user and fully productive?

   _____
   _____
   Other Comment: _____
   _____

4. Are the outputs of the system – especially code highlight actually useful to users?

   _____
   _____
   Other Comment: _____
   _____

5. Do you think there is any weakness of the system? What should we do to improve it?

   _____
   _____
   Other Comment: _____
   _____

Figure 7.1: Sample Survey Form

Questionnaire – Testing on End User

Learning Materials Generating Tools (LMaGTools)

Name	:	Tai Wen Jau
Age	:	22
Field of Study	:	Engeenering
Occupation	:	Student

Please relate the question with the system demonstrated to you.

1. Do you think the user interface is attractive and user friendly?
   Yes, user can understand the interface on first look. Attractive appearance.
   Other Comment: -

2. Is the system easy to use?
   Yes, User don't need too much time to understand how to function the software
   Other Comment: -

3. Are the functions of the system enough for the user and fully productive?
   Yes
   Other Comment: *It would be better if the code simulation part can handle more activity of the code.*

4. Are the outputs of the system – especially code highlight actually useful to users?
   Yes, I think that is the most interesting part of the software. Users can concentrate on one particular part of the code when the highlight is on.
   Other Comment: -

5. Do you think there is any weakness of the system? What should we do to improve it?
   Not really the problem of the software, but I think the setup for this software is quite complicated especially in the customizing PowerPoint part.
   Other Comment: -

Figure 7.2: Selected survey form 1

Questionnaire – Testing on End User

Learning Materials Generating Tools (LMaGTools)

Name	:	Ng Khoon Siah
Age	:	26
Field of Study	:	Software Engeenering
Occupation	:	Senoir Programmer

Please relate the question with the system demonstrated to you.

1. Do you think the user interface is attractive and user friendly?
   Yes, it's easy to handle and following windows standard.
   Other Comment: *Try to let users access some functions like code editor in slide show windows*

2. Is the system easy to use?
   Yes.
   Other Comment: *Try to implement guides on start up of the Add-In, some kind of daily tips.*

3. Are the functions of the system enough for the user and fully productive?
   Yes. I think users can generate good materials from this software
   Other Comment: *Simulation of code is not very accurate, but wouldn't be too far from there.*

4. Are the outputs of the system – especially code highlight actually useful to users?
   Yes, I don't feel bored when I looking to colorful codes, and most important is I can see which part of the code is keyword, which is operator. It's great!
   Other Comment: -

5. Do you think there is any weakness of the system? What should we do to improve it?
   I would like to see more powerful version of code simulation, cause I haven't seen one, except those compilers.
   And also fix some minor bugs in the highlight part, cause I found that when I highlight after editing the code in the box, I have to repeat the action to get accurate result.
   Other Comment: -

Figure 7.3: Selected survey form 2

Figure 7.2 and Figure 7.3 shows two typical answers that provide a lot of help to my evaluation. As for conclusion of the survey, a summary has been done to collect useful comments and suggestions from the end users.

	Strengths	Limitation
User Interface	i. User friendly and attractive ii. Suitable for all user	i. No toolbars in code editor
System Flow	i. Smooth and logical	
System Function	i. Enough and cover user demand ii. Alert user when errors occur iii. Code highlight draw attention to the code	i. Simulator of code is not full function ii. Code highlight may posses minor mistake when dealing with user entered code
System Output	i. Useful and accurate	i. Should let user control path for file output
Other Issues	i. Good timing for presenting with progress bar ii. Does not require huge system resources to handle complicated task	i. Lack of search function in help ii. Setup process takes extra steps to customize PowerPoint iii. Should provide startup tips

7.4 System Strengths

Learning Materials Generating Tools (LMaGTools) has demonstrated the following strength:

i. **Mouse driven**

For the users' sake, LMaGTools is designed in such a way to minimize the keyboard input. Most of the operations are carried out through mouse features and click events. The system will respond accordingly. This will ease the users' task and save a lot of time. Nevertheless, the system does provide keyboard input as alternative to the mouse click for those who used to input with keyboard.

ii. **User friendly**

LMaGTools is user friendly by implementing Graphical User Interface (GUI). The dynamic menu system and graphical toolbars are intuitive. The user interface is designed in consistent manner in order to ease the users' perception and shorten the learning curve, especially to novice users. On the other hand, in order to let the veteran users to apply the system without understanding training, the user interface design is fully conform to Windows standard conditions.

iii. **Effective for teaching programming code**

LMaGTools can easily draw attention of users to the programming code. The code highlight features of the code editor are effectively draw users' attention to concentrate on the highlighted code. Users will easily recognize keywords, operators, and functions calls. Besides that, compare code functions is rather good for indicating difference between two sets of code. By comparing side by side, users can understand what is the lecturer trying to express

iv. **Reduce time needed to generate materials**

By using Office automation features in LMaGTools, time used to generate flexible presentation is greatly reduced. With hidden slide carrying extra information, lecturers can have little extra supportive materials to bring out the lecturing points.

v. **Require little system resource to perform complicated task**

By implementing better programming and coding approach, such as reduce level of hard coding by replacing with control structures and more dynamic way to handle events, system resource used to complete one job is greatly reduce. Thus, reduce chances of system failure.

vi. **Provide extra information to user**

With progress bar on the presentation slide, presenter can clearly see their presentation progress. If it is out of schedule, they will notice the problem sooner.

7.5 System Constraints

Several constraints for this system have been noted. These including:

i. **Simulator of code is not full function**
Simulator of code in code editor is not full developed. It can only apply to one programming language, C or C++. It also lack of ability to handle modern ADE programming, which consist of large amount of three dimensional user interface element such as buttons. Also, it cannot handle, record and manipulate value of variable in the code. This is due to difficulty of resource handling.

ii. **Setup process takes extra steps to customize PowerPoint**
Due to limitation in gaining control of certain PowerPoint property, which had been set to private property, users can only change the value manually. Thus, the setup process takes extra steps to complete.

iii. **No toolbars in code editor**
Users can only access functions in code editor via menu.

7.6 Future Enhancements

Many new ideas bloom while the system is being implemented. However, the time constraint restricts me to incorporate everything into the system. From the very early stage, I have kept on emphasizing system maintainability through exercising modular design. The modularity approach assures low coupling between modules and high cohesion among the internal procedures of a module. Consequently, the extendibility of this system is high.

It is hoped that the following features could be further enhanced in future:

xi. Strengthen features in code simulation. This may be done through reference to external syntax file as what code highlighting does.

xii. Make use of multimedia ideas, animations to upgrade performance and usability of code editor. One feature to consider is to get user comment on certain block of the code while editing, and display it when the explanation of the code reaches the block.

xiii. Enhance more error handling and alert message to avoid users use the Add-In in wrong way.

xiv. Dynamically highlight code while users typing.

7.7 Knowledge and Experience Gained

In short, I had gain intangible knowledge and experience throughout this project. At the preliminary stage, I was trained to gather information and analyze it systematically. During the system and interface design stage, I was given a chance to learn how to design a user friendly, attractive, suitable, and logical user interface.

Moreover, I was exposed to the new terminology such as ActiveX, COM and Add-Ins, which benefit me for future adaptability to new technologies and programming approach. After finished the system development phase, I have learned to test the system by searching the entire system for logical bugs and errors. Although this step sounds easy, it requires a very proper way to conduct the test to avoid carelessness in detecting errors.

The final part is to evaluate the system. To do this, I have learned to conduct an interview with end users and ways to demonstrate the system to them. Besides, I have learned the way to write a report and I have improved my language.

On the other hand, I discover my own weakness in time management. I realized that many factors such as scope and technical implementation should be taking into account precisely in order too deliver the software in time. Finally, I would like to state that this project has unveiled my shortage, reminding me to improve myself and learn more in future.

7.8 Summary

As mentioned earlier, this chapter is the final chapter that concludes the overall system. This chapter included all the problems and solutions occurred during developing the system, system strengths, system limitations, future enhancements, knowledge and experience gained from the project and finally the conclusion of the overall project.

Meanwhile, this chapter shows the way of evaluate the system by end users. Any comment or suggestion from the end users has been put in consideration and should be improve in the future enhancement.

After this chapter is appendix of the project. This part acts as important references for the entire chapters. Some of the important system coding has been attached in the appendix. Besides, the appendix also included the user guide, which guide the user to use the system in proper way.

7.9 Conclusion

Learning Materials Generating Tools (LMaGTools) is a relatively new concept of software that assist presenter, mostly lecturers, to teach programming languages.

LMaGTools is found to be rather practical and useful tool for the lecturers to generate materials for presentation or even helps to handle their lecture presentation. It provides features that automate some Office functions to shorten time spent to edit and create slide. Also, it provides a powerful-featured Code Editor that allows code editing, manipulating, highlighting, comparing, and also simulating.

When comes to presentation, LMaGTools will provide another useful feature – a small progress bar that shows presentation progress and current time in the slide show.

However, there are still a number of future enhancement can be carried out to improve its performance, stability and invest some new development of features. Finally, I believe that LMaGTools will be the first step to encourage later researcher to continue developing it until it is completely functions, reliable and stable.

Bibliography

1. Ari Korhoren and Lauri Malmi, *Algorithm Simulation with Automatic Assessment,* Proceedings of The 4th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'99 pp160 – 163, 1999

2. Tony Greening, Judy Kay, Jeffrey H. Kingston and Kathryn Crawford, *Problem-based learning of first year computer science,* Proceedings of the first Australasian conference on Computer science education, Pages 13 – 18, 1996

3. D. Boud and E. Feletti, *The Challenge of Problem-Based Learning*, Kogan-Page, London (1991).

4. David Boctor, *Microsoft Office 2000 – Visual Basic for Applications Fundamentals*, Washington: Microsoft Press, 1999.

5. Whitten, Bentley, Narlow, *System Analysis And Design Methods*, International Student Edition, Third Edition, Irwin.

6. Pressman, R.S., *Software Engineering: A Practitioner's Approach*, New York: McGraw-Hill, Inc., 1992.

Appendix

Sample Code of PopUpInput.frm

```
Public WithEvents App As Application

Private Sub cmdOk_Click()
   Dim AppEvents As New Application
   Dim lSlideIndex As Long
'
' Check if the message is empty
'
   If txtInput.Text = "" Then
      MsgBox "This is an empty message", vbCritical
   Else
      MsgBox "A message will be insert in the document", vbInformation

'
' Insert new slide for message
'
      lSlideIndex = AppEvents.ActiveWindow.Selection.SlideRange.SlideIndex

      AppEvents.ActiveWindow.View.GotoSlide
Index:=AppEvents.ActivePresentation.Slides.Add(Index:=lSlideIndex + 1,
Layout:=ppLayoutText).SlideIndex

      With AppEvents.ActiveWindow
         If Not .ActivePane.ViewType = ppViewSlide Then
            .Panes(2).Activate
         End If
      End With

      ' insert a new slide for message
      AppEvents.ActiveWindow.View.GotoSlide lSlideIndex + 1

      ' insert the title of the message
      AppEvents.ActiveWindow.Selection.SlideRange.Shapes("Rectangle 2").Select

AppEvents.ActiveWindow.Selection.ShapeRange.TextFrame.TEXTRANGE.Select

AppEvents.ActiveWindow.Selection.ShapeRange.TextFrame.TEXTRANGE.Charact
ers(Start:=1, Length:=0).Select
      With AppEvents.ActiveWindow.Selection.TEXTRANGE
         .Text = cboType.Text
         With .Font
            .Name = "Lucida Sans Unicode"
            .Size = 40
            .Bold = msoFalse
            .Italic = msoFalse
            .Underline = msoFalse
```

```
            .Shadow = msoFalse
            .Emboss = msoFalse
            .BaselineOffset = 0
            .AutoRotateNumbers = msoFalse
            .color.SchemeColor = ppTitle
        End With
    End With

    ' Change display style of the title shape
    With AppEvents.ActiveWindow.Selection.ShapeRange
        .Fill.Visible = msoTrue
        .Fill.Solid
        .Fill.ForeColor.RGB = RGB(255, 255, 153)
        .Fill.Transparency = 0.5
        .Line.Weight = 4.5
        .Line.Style = msoLineThickThin
        .Line.Visible = msoTrue
        .Line.ForeColor.RGB = RGB(51, 51, 153)
        .Line.BackColor.RGB = RGB(255, 255, 255)
    End With

    ' insert message
    AppEvents.ActiveWindow.Selection.SlideRange.Shapes("Rectangle 3").Select

AppEvents.ActiveWindow.Selection.ShapeRange.TextFrame.TEXTRANGE.Select

AppEvents.ActiveWindow.Selection.ShapeRange.TextFrame.TEXTRANGE.Charact
ers(Start:=1, Length:=0).Select
    With AppEvents.ActiveWindow.Selection.TEXTRANGE
        .Text = txtInput.Text
        With .Font
            .Name = "Lucida Sans Unicode"
            .Size = 32
            .Bold = msoFalse
            .Italic = msoFalse
            .Underline = msoFalse
            .Shadow = msoFalse
            .Emboss = msoFalse
            .BaselineOffset = 0
            .AutoRotateNumbers = msoFalse
            .color.RGB = RGB(Red:=0, Green:=51, Blue:=102)
        End With
    End With

    With AppEvents.ActiveWindow.Selection.ShapeRange
        .Fill.Visible = msoTrue
        .Fill.Solid
        .Fill.ForeColor.RGB = RGB(255, 255, 153)
        .Fill.Transparency = 0.5
        .Line.Weight = 4.5
```

```
        .Line.Style = msoLineThickThin
        .Line.Visible = msoTrue
        .Line.ForeColor.RGB = RGB(51, 51, 153)
        .Line.BackColor.RGB = RGB(255, 255, 255)
    End With

'
' Insert action button to go back
'

HostApp.ActiveWindow.Selection.SlideRange.Shapes.AddShape(msoShapeActionB
uttonBackorPrevious, 96#, 482#, 42#, 36#).Select
    With
HostApp.ActiveWindow.Selection.ShapeRange.ActionSettings(ppMouseClick)
        .Action = ppActionLastSlideViewed
        .SoundEffect.Type = ppSoundNone
        .AnimateAction = msoTrue
    End With
    With
HostApp.ActiveWindow.Selection.ShapeRange.ActionSettings(ppMouseOver)
        .Action = ppActionNone
        .SoundEffect.Type = ppSoundNone
        .AnimateAction = msoFalse
    End With
    With HostApp.ActiveWindow.Selection.ShapeRange
        .Fill.Visible = msoTrue
        .Fill.Solid
        .Fill.ForeColor.SchemeColor = ppAccent1
        .Line.Visible = msoFalse
    End With

'
' Hide the message slide
'
    AppEvents.ActiveWindow.Selection.SlideRange.Name = cboType.Text
    AppEvents.ActiveWindow.Selection.SlideRange.SlideShowTransition.Hidden =
msoTrue
    AppEvents.ActiveWindow.Selection.Unselect

'
' Back to slide where the procedure called
'
    HostApp.ActiveWindow.View.GotoSlide lSlideIndex

'
' Insert action button to show message
'

HostApp.ActiveWindow.Selection.SlideRange.Shapes.AddShape(msoShapeActionB
uttonInformation, 54#, 482#, 42#, 36#).Select
```

```
      With
HostApp.ActiveWindow.Selection.ShapeRange.ActionSettings(ppMouseClick)

        With .Hyperlink
          .Address = ""
          .SubAddress = cboType.Text
        End With
      End With
      With
HostApp.ActiveWindow.Selection.ShapeRange.ActionSettings(ppMouseOver)
        .Action = ppActionNone
        .SoundEffect.Type = ppSoundNone
        .AnimateAction = msoFalse
      End With
      With HostApp.ActiveWindow.Selection.ShapeRange
        .Fill.Visible = msoTrue
        .Fill.Solid
        .Fill.ForeColor.SchemeColor = ppAccent1
        .Line.Visible = msoFalse
      End With

'
' Unload form
'
      Unload Me
    End If
End Sub

'
' Unload form
'
Private Sub cmdCancel_Click()
    Unload Me
End Sub
```

Questionnaire – Testing on End User
Learning Materials Generating Tools (LMaGTools)

Name : TAI WEN JIAN
Age : 22
Field of Study : Engineering
Occupation : Student

Please relate the question with the system demonstrated to you.

1. Do you think the user interface is attractive and user friendly?
Yes, user can understand the interface on first look. Attractive appearance
Other Comment: -

2. Is the system easy to use?
Yes, user don't need too much time to understand how to function the software
Other Comment: -

3. Are the functions of the system enough for the user and fully productive?
yes.
Other Comment: It would be better if the code simulation part can handle more activity of the code

4. Are the outputs of the system – especially code highlight actually useful to users?
yes, I think that's the most interesting part of the software. users can concentrate on on particular part of the code when the highlight is on
Other Comment: -

5. Do you think there is any weakness of the system? What should we do to improve it?
Not really the problem of the software, but I think the setup for this software is quite complicated especially in the customizing ppoint part.
Other Comment: -

Questionnaire – Testing on End User
Learning Materials Generating Tools (LMaGTools)

Name : Ng Khoon Siah
Age : 26
Field of Study : Software Engineering
Occupation : Senior Programmer

Please relate the question with the system demonstrated to you.

1. Do you think the user interface is attractive and user friendly?
Yes, it's easy to handle and it's following windows standard.
Other Comment: Try to let users access some functions like code editor in Hide show windows

2. Is the system easy to use?
yes.
Other Comment: Try to implement guides on startup of the Add-In, some kind of dayily tips.

3. Are the functions of the system enough for the user and fully productive?
Yes. I think users can generate good materials from this software
Other Comment: Simulation of code is not very accurate, but wouldn't be too far from there.

4. Are the outputs of the system – especially code highlight actually useful to users?
Yes, I don't feel bored when I looking to colorful codes, and most important is I can see which part of the code
Other Comment: is keyword, which is operator. It's great!

5. Do you think there is any weakness of the system? What should we do to improve it?
I would like to see more powerful version of code simulation, cause I haven't seen one, except those compilers. And also
Other Comment: fix some minor bugs in the highlight code part, cause I found that when I highlight after editting the code in the box, I have to repeat the action to get accurate result.

User Manual

About LMaGTools

LMaGTools is a Microsoft PowerPoint COM Add-in being carried out to fulfill requirements of the final year project, WXES 3182: Latihan Ilmiah Tahap Akhir II, as part of the Bachelor of Science Computer Degree of the Faculty of Computer Science and Information Technology, University of Malaya.

This project had been supervised by Ms. Nor Aniza and moderated by Mr. Ibrahim Abu Bakar.

Law Yong Sein

What is LMaGTools

LMaGTools (Learning Materials Generating Tools) is a COM Add-In being designed to work with Microsoft PowerPoint 2000. It provides additional functions to Microsoft PowerPoint 2000 to enrich the existing presentation environment. These functions are designed for Learning-Material-Generating for programming language courses.

LMaGTools provides an intuitive and user-friendly interface. Every function provided can be access via collected menu and toolbars.

The main idea of LMaGTools is to enable features required to generate and present materials to conduct a programming language lecture. Thus, besides some additional features to help generating these materials, LMaGTools also provide features to explain programming codes.

For better timing during presentation, a progress bar placed in the slide window enable lecturers to know current slide progress and running time.

Table of Content

ABOUT LMAGTOOLS I

WHAT IS LMAGTOOLS II

TABLE OF CONTENT III

TABLE OF FIGURE IV

CHAPTER 1 GETTING STARTED 1

1.1 SYSTEM REQUIREMENTS 1

1.2 INSTALLATION 1

1.2.1 LMaGTools' Setup 1

1.2.2 Customize MS PowerPoint 2

1.2.3 Connect and Disconnect from PowerPoint 2000 5

CHAPTER 2 ELEMENTS IN LMAGTOOLS 7

2.1 MENU AND TOOLBARS 7

2.1.1 LMaGTools' Menu 7

2.1.2 LMaGTools' Toolbar 8

2.2 INSERT MESSAGE (HIDDEN SLIDE) 8

2.2.1 Insert Message - How It Works 8

2.2.2 The Input Interface 9

2.2.3 Display the Message 10

2.3 CODE EDITOR 11

2.3.1 Code Editor Interface 11

2.3.2 Menu Section 12

2.3.3 Code Section 16

2.3.4 Control Section 17

2.3.5 System Message Area 18

2.4 CONVERT FILE FORMAT 19

2.4.1 How Converting Works 19

2.4.2 Converting Interface 19

2.5 SLIDE SHOW PROGRESS BAR 20

2.5.1 Progress Bar 20

List of Figures

FIGURE 1.1: INITIAL DIALOG BOX FROM SETUP EXECUTABLE FILE 1

FIGURE 1.2: DIALOG BOX THAT ALLOW USERS TO SET INSTALLATION PATH 2

FIGURE 1.3: LOCATION OF <OPTION...> BUTTON 3

FIGURE 1.4: UNCHECK THE <END WITH BLACK SLIDE> OPTION 4

FIGURE 1.5: LOCATION OF <CUSTOMIZE...> BUTTON 4

FIGURE 1.6: ADD "COM ADD-INS" BUTTON TO MENU BAR 5

FIGURE 1.7: "COM ADD-INS" BUTTON IN POWERPOINT MENU BAR 5

FIGURE 1.8: CONNECT LMAGTOOLS 6

FIGURE 2.1: MENU ITEMS FOR LMAGTOOLS 7

FIGURE 2.2: LMAGTOOLS TOOLBARS 8

FIGURE 2.3: INPUT INTERFACE FOR INSERT MESSAGE FUNCTION 9

FIGURE 2.4: <INFORMATION> ACTION BUTTON 10

FIGURE 2.5: A HIDDEN SLIDE WITH <BACK> ACTION BUTTON 11

FIGURE 2.6: AN INITIAL INTERFACE OF CODE EDITOR IN LMAGTOOLS 12

FIGURE 2.7: MENU OF CODE EDITOR 12

FIGURE 2.8: FILE MANIPULATION FUNCTIONS 13

FIGURE 2.9: HIGHLIGHT CODE FUNCTIONS 14

FIGURE 2.10: SET SYNTAX POP UP 14

FIGURE 2.11: COMPARE CODE FUNCTIONS 14

FIGURE 2.12: SPLIT CODE FUNCTIONS 15

FIGURE 2.13: SHELL OPEN PRE-INSTALLED COMPILER FUNCTION 15

FIGURE 2.14: HORIZONTALLY SPLIT CODE SECTION 16

FIGURE 2.15: VERTICALLY SPLIT CODE SECTION 17

FIGURE 2.16: CONTROL SECTION OF CODE EDITOR 17

FIGURE 2.17: SYSTEM MESSAGE AREA OF CODE EDITOR 18

FIGURE 2.18: SYSTEM MESSAGE REQUIRES INPUT WITH AN INPUT BOX 18

FIGURE 2.19: CONVERT FUNCTION POP UP 19

FIGURE 2.20: SLIDE SHOW PROGRESS BAR 20

Chapter 1 Getting Started

1.1 System Requirements

1. Windows 95/98
2. Microsoft Office 2000
3. 64 MB RAM (Recommended)
4. Turbo C++ 3.0 Compiler (Optional)

1.2 Installation

LMaGTools has come with a simple-to-operate Setup Package. With this Setup Package, you can install LMaGTools in your PC in steps, as stated below:

1.2.1 LMaGTools' Setup

To install LMaGTools in your PC, run executable file "Setup.exe" from media containing setup information (Attached CD-R).

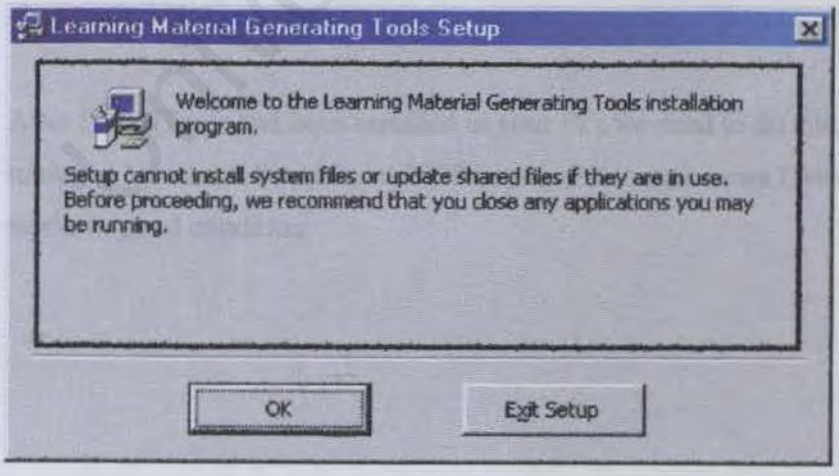

Figure 1.1: Initial dialog box from setup executable file

From first dialog box, choose <OK> button to continue setup, else choose <Exit Setup> button to quit the process.

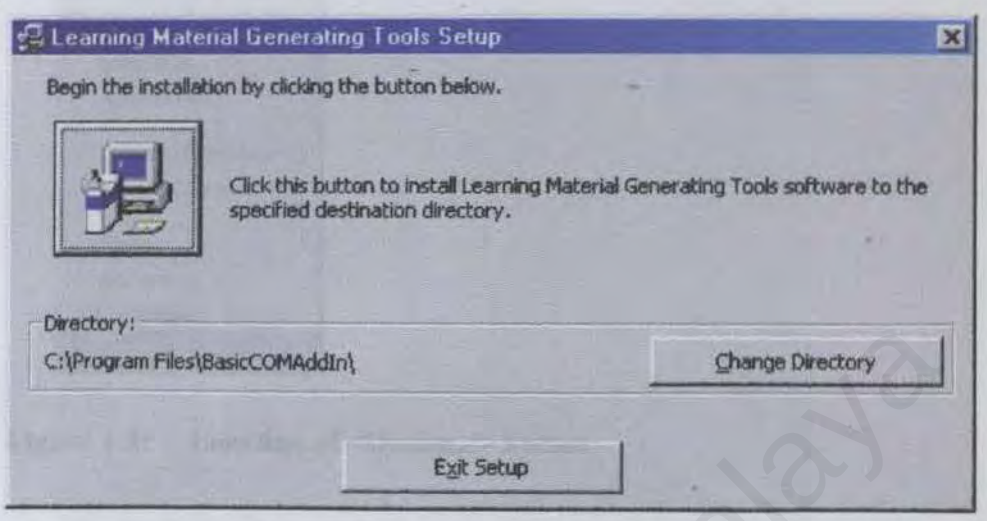

Figure 1.2: Dialog box that allow users to set installation path

In the following dialog box, select the iconic button to start install LMaGTools in the default directory path. If you want to choose any other installation path, click <Change Directory> button to choose or create desired directory. If you wish to stop the installation process, click <Exit Setup> button to quit.

1.2.2 Customize MS PowerPoint

After LMaGTools had been installed in your PC, we need to do a little fine-tuning in Microsoft PowerPoint 2000 in your PC to make sure LMaGTools works in good condition.

1. Click the <Option...> button under <Tools> menu button to bring up the Option Panel.

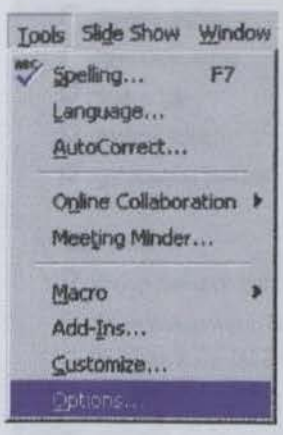

Figure 1.3: Location of <Option...> button

2. In the Option Panel, uncheck the <End with black slide> option to prevent inappropriate slide number reported to the progress bar while showing the black slide. If this checkbox is check, and illegal message will generate and LMaGTools will be forced to stop from normal operations. After this, click <OK> button to verify the change.

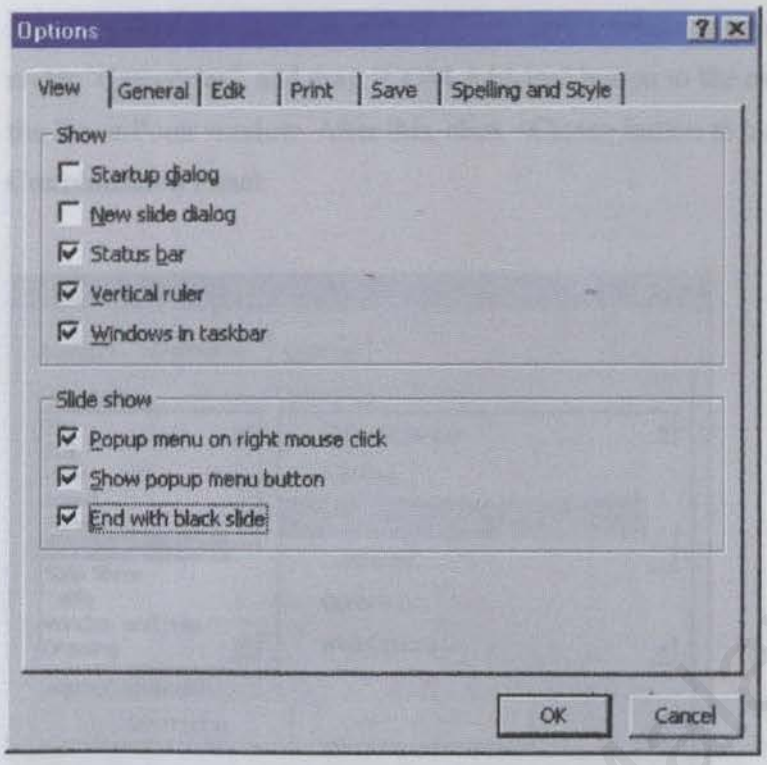

Figure 1.4: Uncheck the <End with black slide> option

3. Then, click <Customize...> button under <Tools> menu button to bring up the Customization Panel.

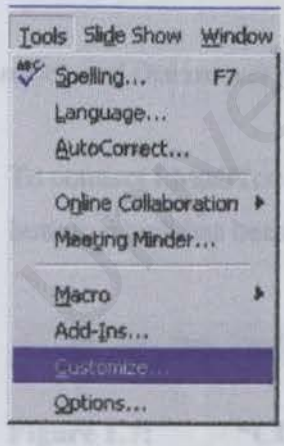

Figure 1.5: Location of <Customize...> button

4. In the Customization Panel, select <Commands> tab, and select "Tools" under "Categories", and drag "COM Add-Ins" button to the menu bar of the PowerPoint window. After this, click <Close> button to close the Customization Panel.

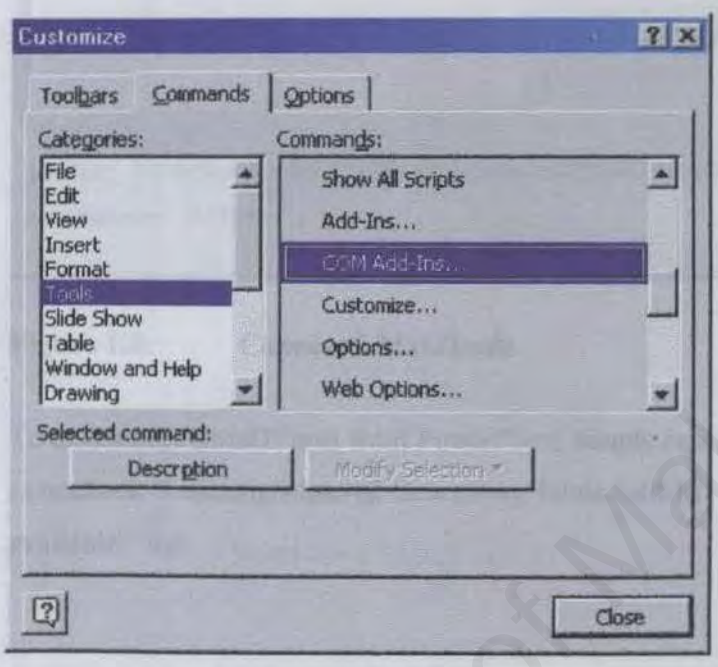

Figure 1.6: Add "COM Add-Ins" button to menu bar

1.2.3 Connect and Disconnect from PowerPoint 2000

1. To connect LMaGTools to PowerPoint, click on the <COM Add-Ins> button, which has been drag out from the Customization Panel.

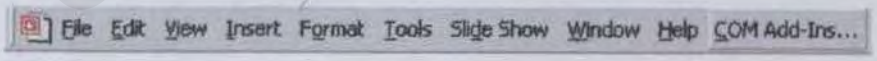

Figure 1.7: "COM Add-Ins" button in PowerPoint menu bar

2. In the COM Add-Ins dialog box, check "Learning Material Generating Tools Add-In" in "Add-Ins available:" list. Click <OK> button to connect LMaGTools.

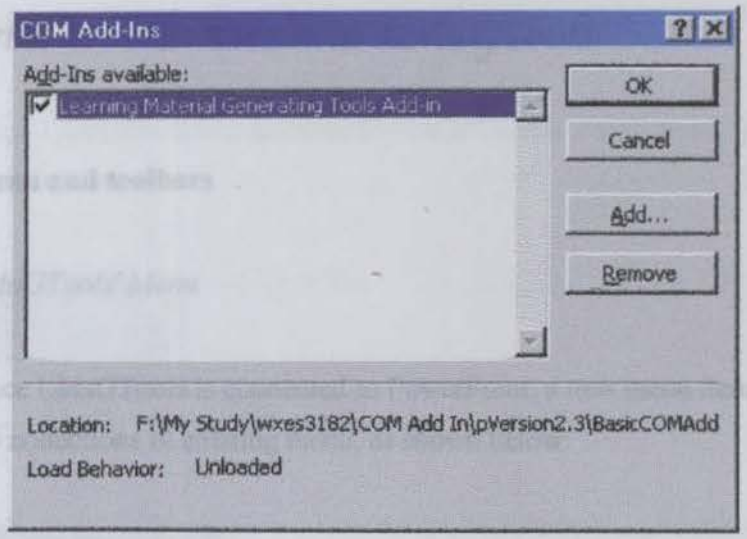

Figure 1.8: Connect LMaGTools

3. To disconnect LMaGTools from PowerPoint, simply repeat steps 5 and 6 to uncheck "Learning Material Generating Tools Add-In" in "Add-Ins available:" list.

Chapter 2 Elements in LMaGTools

2.1 Menu and toolbars

2.1.1 LMaGTools' Menu

Once LMaGTools is connected to PowerPoint, a new menu item is added to the collections of existing menu, as shown below:

Figure 2.1: Menu items for LMaGTools

Menu for LMaGTools consist of four buttons in total. They are:

1. *<Insert Message...> button*
   Bring up input dialog box for users to enter their message
2. *<Display Code...> button*
   Bring up code editor for code displaying, editing and other actions regarding codes explanation.
3. *<Convert to...> button*
   Bring up dialog box for users to choose a format of desired output of the converted document.
4. *<Help> button*
   Display user guide for LMaGTools.

2.1.2 *LMaGTools' Toolbar*

Once LMaGTools is connected to PowerPoint, a floating toolbar is added to it, as shown below:

Figure 2.2: LMaGTools toolbars

Toolbar for LMaGTools consist of four buttons in total, which are similar to button in LMaGTools menu. They are:

1. 
   Bring up input dialog box for users to enter their message

2. 
   Bring up code editor for code displaying, editing and other actions regarding codes explanation.

3. 
   Bring up dialog box for users to choose a format of desired output of the converted document.

4. 
   Display user guide for LMaGTools.

2.2 Insert Message (Hidden Slide)

2.2.1 *Insert Message - How It Works*

This feature from LMaGTools is an Office automating features, which simplified work to create a hidden message slide in PowerPoint. Steps it takes to complete the task are:

1. Getting input from user from the input interface.
2. Create a new slide after current slide, where this function was called.
3. Write "Title" and "Message" to the new slide based on information from input interface.
4. Add an action button link to the slide, which shown before the message slide.
5. Hide the slide, so it would not be shown if it is not needed in the presentation.
6. Add an action button in the current slide, where this function was called. Set the link of the action button to the hidden slide.

By using this function, these steps can be completed with only entering "Message Title" and "Message Text" through the input interface. And thus, time is saved for better use. Besides that, a unified layout makes user easy to understand that this is a message and the whole presentation look tidy.

2.2.2 *The Input Interface*

When the <Insert Message> function is called, a dialog box, which allow users to input their message will pop up as shown:

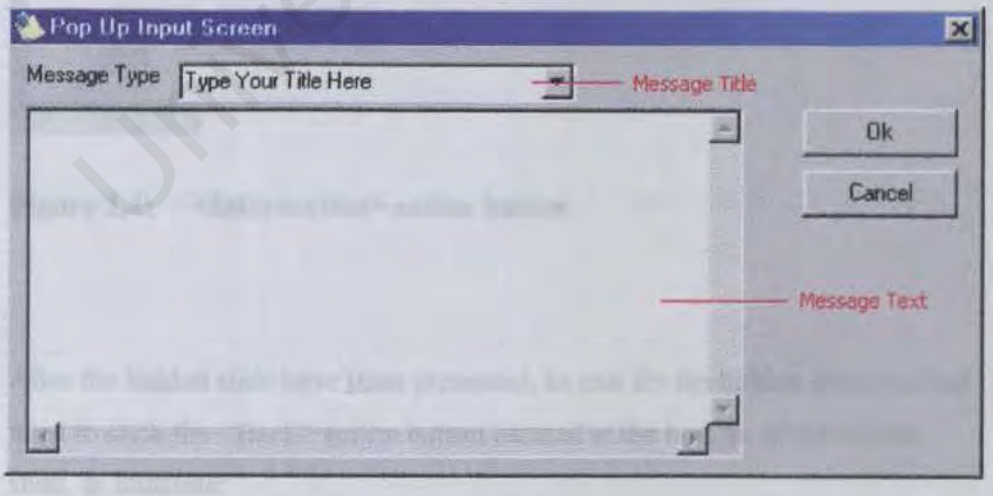

Figure 2.3: Input interface for Insert Message function.

You can type your message title in the "Message Title" combo box, or simply choose one frequently used title from the drop down. These frequently used title include:

- Good Programming Practice
- Software Engineering Observations
- Performance Tips
- Portability Tips
- Look-and-Feel Observations
- Testing and Debugging Tips
- Common Programming Errors

Then you can input your message in the "Message Text" textbox. After that, click <OK> button to insert this message to the presentation.

2.2.3 *Display the Message*

To display the message while presentation, you just need to click the <Information> action button at the bottom of the slide, as illustrates:

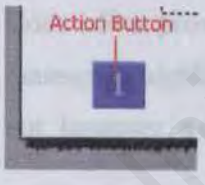

Figure 2.4: <Information> action button

After the hidden slide have been presented, to exit fro the hidden slide you just need to click the <Back> action button located at the bottom of the hidden slide, as illustrate:

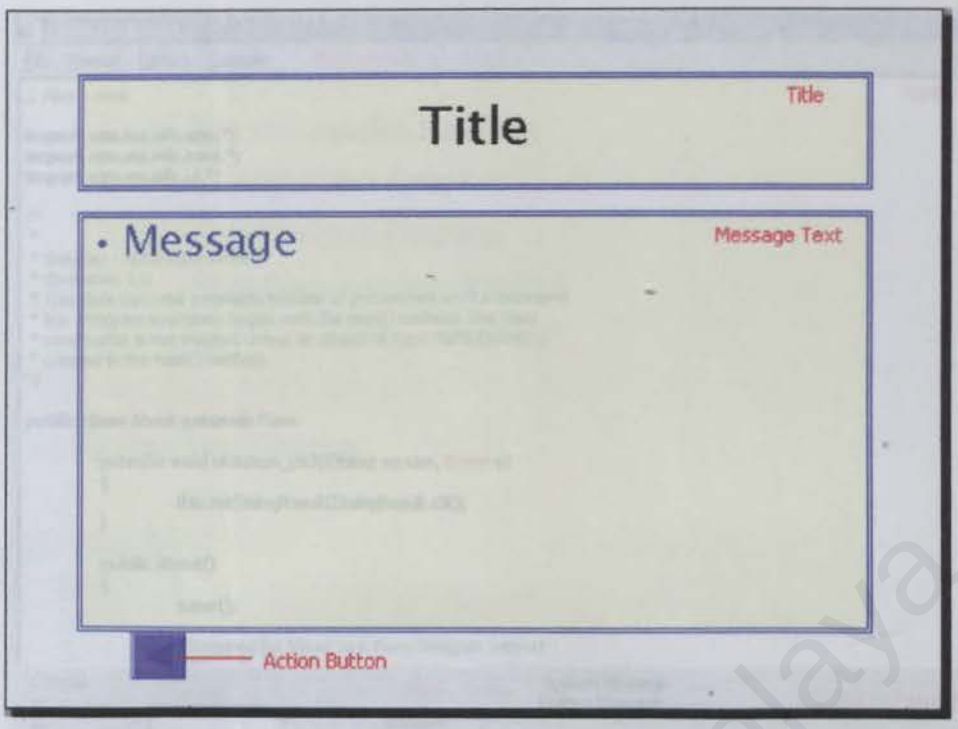

Figure 2.5: A hidden slide with <Back> action button

2.3 Code Editor

2.3.1 Code Editor Interface

Code editor provided by LMaGTools is a tool that allows user to edit, managing, highlighting and do other formatting to programming codes. It is not, however, a compiler or an interpreter.

The editor's interface is simple and compact to be user friendly, as illustrated:

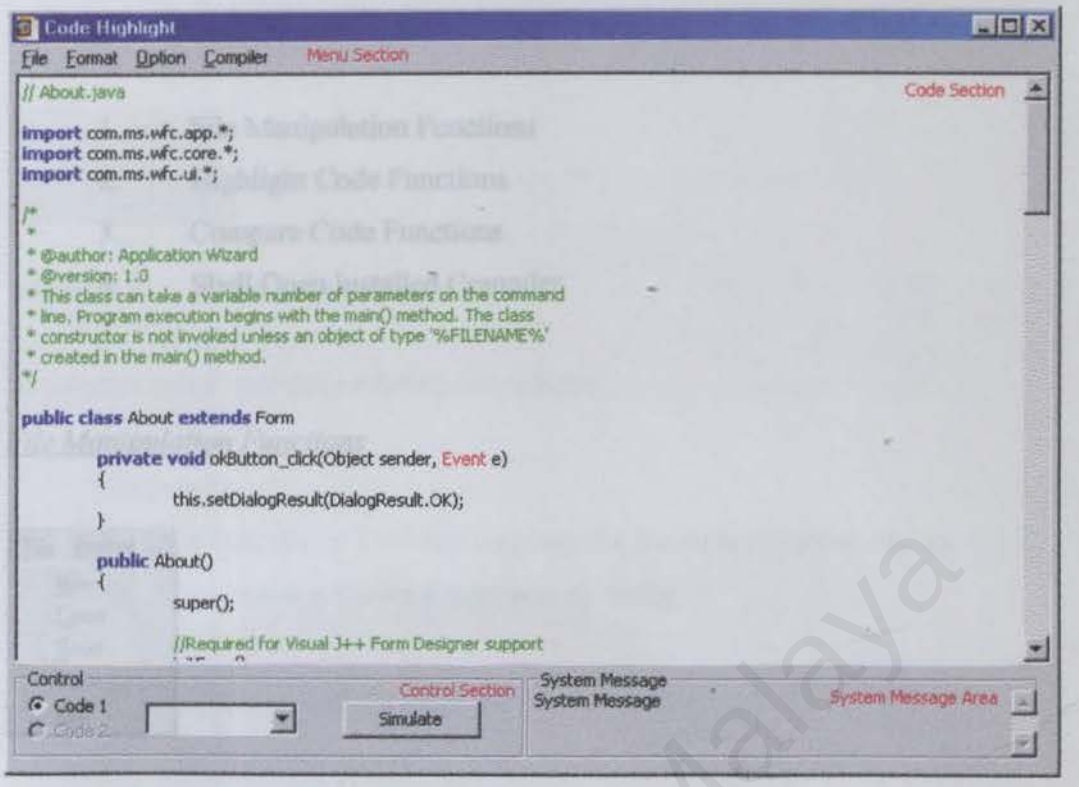

Figure 2.6: An initial interface of Code Editor in LMaGTools

There are four major sections in the user interface:

1. Menu Section
2. Code Section
3. Control Section
4. System Message Area

2.3.2 Menu Section

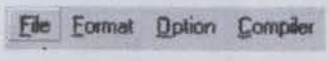

Figure 2.7: Menu of Code Editor

Menu of LMaGTools' Code Editor consist of four group of functional button:

1. File Manipulation Functions
2. Highlight Code Functions
3. Compare Code Functions
4. Shell Open Installed Compiler

File Manipulation Functions

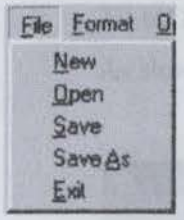

Figure 2.8: File manipulation functions

There are five functions provided by File Manipulation Functions:

1. *New*
   To close active code being edited and open a white page.
2. *Open*
   To close active code being edited and open an existing file by browsing file name through the pop up dialog box.
3. *Save*
   To save active code being edited.
4. *Save As*
   To save active code being edited to another file.
5. *Exit*
   To close the editor and return to PowerPoint environment.

Highlight Code Functions

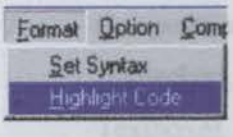

Figure 2.9: Highlight code functions

Highlight Code Functions including two actions:

1. *Set Syntax*

   To set syntax file by selecting language for the code in pop up dialog. As shown below is the set syntax pop up dialog:

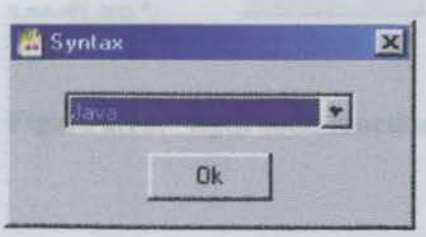

Figure 2.10: Set syntax pop up

2. *Highlight Code*

   To highlight the edited text in code section by referencing syntax file selected in "Set Syntax" function.

Compare Code Functions

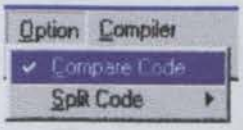

Figure 2.11: Compare code functions

Compare Code Functions have two actions:

1. *Compare code*

   If checked, code section will divide into two independent Rich Textboxes.

2. *Split Code*

   To set whether to split the code section vertically or horizontally. As illustrated:

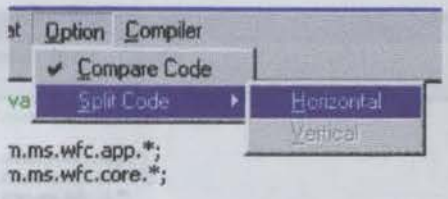

Figure 2.12: Split code functions

Shell Open Installed Compiler

Figure 2.13: Shell open pre-installed compiler function

This function opens an existing external programming code compiler, such as Turbo C++. However, the path of the compiler must be the default setup path, or the shell open function would not able to execute the compiler.

2.3.3 *Code Section*

Code section of the Code Editor is a Customized Rich Textbox, which allows users to type, edit, format and highlight (colour) their code. In normal mode, there will be only one Rich Textbox appear in the code section. When user toggles the compare code function, another Rich Textbox will be located in the code section. There are two ways these Rich Textbox been placed.

1. Horizontally, as shown:

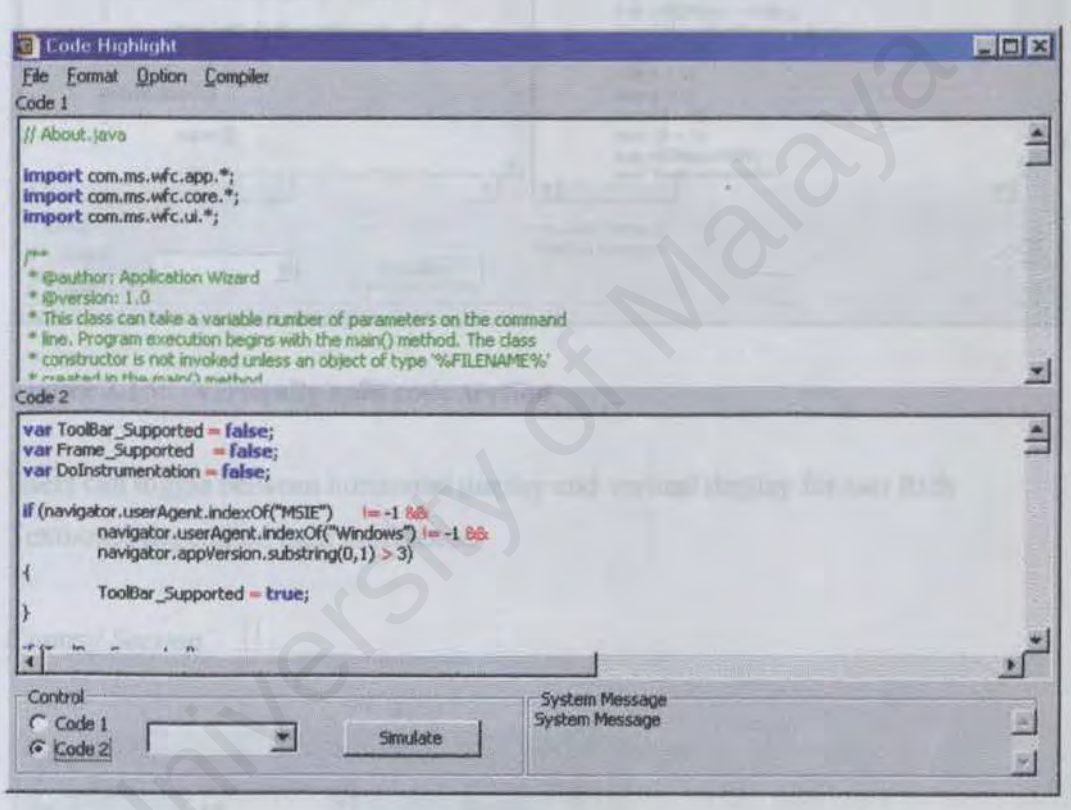

Figure 2.14: Horizontally split code section

2. Vertically, as shown:

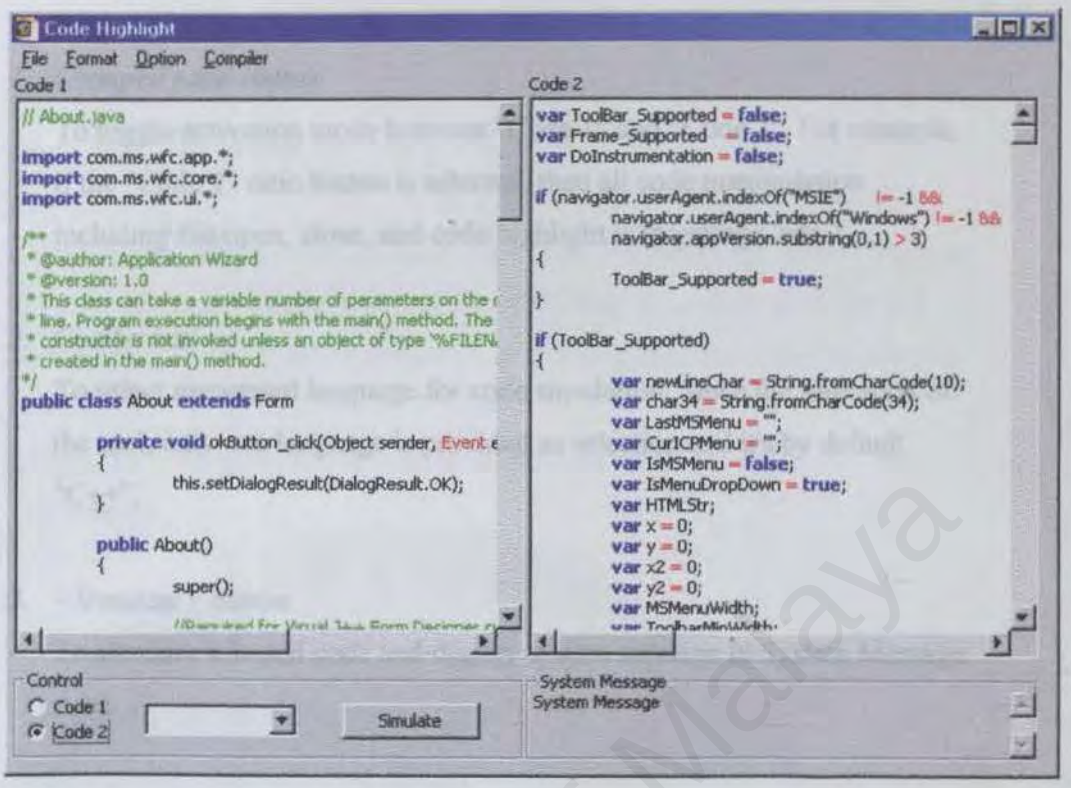

Figure 2.15: Vertically split code section

Users can toggle between horizontal display and vertical display for two Rich Textboxes depend on their preference.

2.3.4 Control Section

Figure 2.16: Control Section of Code Editor

The control section consist of three different sets of controls:

1. *Grouped ratio button*

   To toggle activation mode between "Code 1" and "Code 2". For example, if the "Code 1" ratio button is selected, then all code manipulation including file open, close, and code highlight is targeting Code 1.

2. *Combo box*

   To select associated language for code simulation. Since the limitation of the tool, only one language is provided as selection and it's by default "C++".

3. *<Simulate> button*

   To simulate selected code and display system message in System Message Area.

2.3.5 *System Message Area*

Figure 2.17: System message area of Code Editor

System message area is the area where system message will be displayed when selected code in Code Section is being simulated. If the simulation requires input from users, an input box will be displayed, as shown:

Figure 2.18: System message requires input with an input box

2.4 Convert File Format

2.4.1 How Converting Works

Conversion function in LMaGTools is function that inheriting <save> file type in PowerPoint. By selecting desired format of output for converted file in a popup dialog box, the active presentation will be converted to format selected.

Output file is set to be placed in default directory of all documents, "C:\My Documents\".

2.4.2 Converting Interface

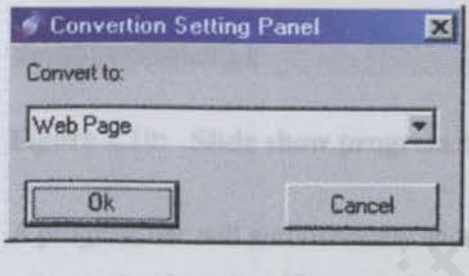

Figure 2.19: Convert function pop up

Converting interface is very simple. Users just have to choose the file format they want to convert from the combo box, and then click <Ok> button and everything will be done. Converted file will be placed in the directory "C:\My Documents\".

There are total of seven formats provided in the combo box:

1. Web Page
2. Rich Text File
3. GIF Graphics Interchange Format
4. JPEG File Interchange Format
5. PNG Portable Network Graphics Format
6. TIFF Tag Image File Format
7. Device Independent Bitmap

2.5 Slide Show Progress Bar

2.5.1 Progress Bar

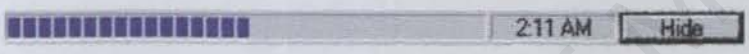

Figure 2.20: Slide show progress bar

A progress bar will automatically shown in slide show window while presentation is active. There are 3 parts in this progress bar:

1. *Progress Indicator*
Shows current progress of the slide show

2. *Current Time*
Shows current time

3 *<Hide> Button*
To hide the progress bar

The progress bar is programmed to enable users to move it around by click on any part of the bar, and drag it to a new position.